LEARNING TO ADAPT WITHOUT GRADIENTS: EMERGENCE OF MENTAL REPRESENTATIONS VIA SECOND-ORDER LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Mental representation, characterized by structured internal models that mirror external environments, is fundamental to advanced cognition but remains challenging to investigate empirically. Existing theory suggests that second-order learning mechanisms that adapt first-order learning, i.e., learning about the underlying task or domain, can promote the emergence of environment–cognition isomorphism. In this work, we empirically validate this hypothesis using a hierarchical framework with a Graph Convolutional Network (GCN) as a first-order learner and an MLP-based controller as a second-order learner. The controller dynamically adapts the GCN's parameters in response to structurally novel environments, enabling fast adaptation without gradient-based optimization. Across synthetic maze tasks and real-world road network graphs, we show that second-order learning improves generalization under distributional shift. Notably, the approach achieves performance comparable to gradient-based meta-learning methods such as MAML, while requiring no test-time gradients or labeled data and performing adaptation in a single forward pass. We further show that effective adaptation is linked to the emergence of structured latent representations that are isomorphic to the environment; when this isomorphism is disrupted, performance degrades significantly. These results provide empirical evidence for the role of structured representations in second-order learning and establish a practical alternative to gradient-based meta-learning for graph-structured environments.

1 INTRODUCTION

Understanding the computational mechanisms behind mental representation the formation of internal cognitive structures aligned with environmental structures is crucial for advancing artificial cognitive systems. Historically, theories such as cognitive maps proposed by Tolman Tolman (1948) have offered a framework to conceptualize these representations, emphasizing their role in enabling flexible navigation and efficient problem-solving. Arnold et al. Arnold et al. (2012; 2015) proposed a theoretical model suggesting that such representational cognition emerges from evolutionary pressures favoring second-order learning, which involves mechanisms adapting first-order learning itself. In their framework, selection pressure on second-order learning results in selection pressure on isomorphism-based execution of first-order learning. Mental representation constitutes a component of this isomorphism. The hypothesis that mental representation is a fundamental aspect of advanced cognition aligns with neuroscientific findings showing that the brain develops internal spatial representations isomorphic to environmental structure. For instance, studies on place and grid cells suggest that "place cells are the basic elements of a distributed non-centered map-like representation," enabling organisms to perceive and remember their position in space Moser et al. (2008). These findings support the notion that internal cognitive structures (i.e., mental representations) emerge from systems that structurally mirror the external world. Despite the theoretical appeal, there is limited direct empirical evidence demonstrating how second-order learning induces first-order learning mechanisms that are structurally aligned with the environment. Furthermore, most existing computational approaches rely explicitly on evolutionary algorithms, which, while powerful, can be computationally demanding and less interpretable in terms of learning dynamics.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

In this paper, we investigate this hypothesis using a hierarchical learning framework that combines Graph Convolutional Networks (GCNs) and second-order adaptive controllers implemented through Multi-Layer Perceptrons (MLPs). Specifically, the GCN serves as a first-order learner that predicts optimal paths in maze navigation tasks, while an MLP-based second-order learner dynamically adjusts the GCN's parameters in response to structurally novel environments. While our model is trained using standard gradient-based optimization, we use the term "without gradients" to refer specifically to test-time adaptation: the model adapts via a single forward pass without requiring gradient computation or labeled data during inference. We show that second-order learning improves the adaptive capacity of the network. Moreover, its effectiveness is enhanced when an internal mental map structurally isomorphic to the environment emerges within the latent representations.

Our contributions are summarized as follows:

1. We propose a hierarchical second-order learning (SOL) framework that combines a Graph Convolutional Network (GCN) with an MLP-based controller for adaptive reasoning in graph-structured environments. The controller predicts task-conditioned parameter updates to a pre-trained model conditioned on its internal representations, enabling efficient adaptation without gradient-based optimization at test time.
2. We provide empirical evidence supporting Arnold et al.'s Arnold et al. (2012) theoretical prediction by showing that second-order learning is associated with the emergence of structured latent representations that are isomorphic to the environment, quantitatively supported by high correlation metrics.
3. We demonstrate robust out-of-distribution generalization across synthetic maze environments under increasing structural perturbations, where the proposed approach significantly improves adaptation performance over unadapted models and maintains stable accuracy.
4. We show that the proposed method achieves performance comparable to gradient-based meta-learning approaches such as MAML, while requiring no test-time gradients or labeled data and performing adaptation via a single forward pass, resulting in substantial computational efficiency gains.
5. We validate the approach on real-world road network graphs derived from OpenStreetMap (OSM), using sampled subgraphs to simulate dynamic navigation scenarios, demonstrating that second-order learning generalizes beyond synthetic environments under realistic structural perturbations.

Our work thus bridges a critical gap between theoretical predictions and practical neural network implementations, providing concrete computational insights into the evolution and mechanisms underlying mental representation formation.

2 Background and Related Work

Mental Representation and Cognitive Maps. Cognitive maps describe internal representations that preserve the structure of external environments, enabling flexible navigation and reasoning Tolman (1948); Banino et al. (2018); Cueva & Wei (2018). Recent work shows that spatially structured and distance-preserving representations can emerge in neural models of navigation and spatial representation Wijmans et al. (2023); Dehong et al. (2025), while graph-based models capture spatial and relational structure in learned representations Wu et al. (2021); Battaglia et al. (2018). However, prior work primarily focuses on representation learning itself, rather than explaining how such structured representations emerge from adaptive learning mechanisms.

Second-Order Learning and Meta-Learning. Second-order learning refers to processes that adapt first-order learning, as illustrated in learning-to-learn paradigms Harlow (1949). Modern meta-learning methods such as MAML and Reptile Finn et al. (2017); Nichol et al. (2018) operationalize this idea through gradient-based adaptation, enabling rapid generalization across tasks. While effective, these approaches require iterative optimization, test-time gradients, and often labeled data. Arnold et al. Arnold et al. (2012; 2015) further hypothesize that second-order learning induces selection pressure for environment-aligned, isomorphic representations, but this prediction lacks direct empirical validation.

Graph Neural Networks and Adaptive Models. Graph neural networks (GNNs), particularly GCNs Kipf & Welling (2017); Veličković et al. (2017), are well-suited for modeling structured environments and have been widely applied to navigation and relational reasoning tasks. Recent work combining GNNs with meta-learning improves adaptability Jiang et al. (2019); Zügner & Günnemann (2019), but these approaches typically rely on gradient-based updates and do not explicitly examine how adaptation mechanisms influence the structure of learned representations.

Hypernetworks and Adaptive Weight Generation. Hypernetworks generate task-conditioned model parameters Ha et al. (2017), with roots in approaches such as HyperNEAT Stanley et al. (2009), but often produce full model weights and rely on gradient-based training. MotherNet generates task-specific child networks in a single forward pass from labeled training sets Müller et al. (2025). In contrast, SOL predicts targeted updates for a pre-trained model from its outputs and parameters, enabling label-free, gradient-free test-time adaptation in a single forward pass.

Our Positioning and Novelty. While cognitive maps, second-order learning, adaptive GNNs, and hypernetworks have been studied extensively, there is limited empirical investigation of whether structured, environment-aligned representations are related to the ability of a learned second-order mechanism to adapt a first-order model under distribution shift. We study this question through a framework in which the state of a trained first-order graph learner is mapped directly to a task-conditioned reparameterization, enabling adaptation without test-time gradients or labeled data. Across structural and reward perturbations, we evaluate both adaptation behavior and the organization of the first-order representation, providing a computational test of the relationship proposed by Arnold et al. Arnold et al. (2012).

3 METHODOLOGY

Our framework investigates hierarchical learning through a combination of first-order and second-order learning mechanisms, specifically within a maze navigation task. Our primary aim is to empirically demonstrate how second-order learning facilitates the emergence of structured internal representations that align closely (are isomorphic) with the external maze environment.

We consider maze navigation tasks formulated as node classification problems, where a graph-based representation of the maze is given by $G = (V, E)$, with nodes $v \in V$ having associated features such as spatial coordinates, blockage information, and degrees of connectivity. The objective is to classify each node as being part of the shortest path from a start to a goal node.

First-order learning is implemented using a Graph Convolutional Network (GCN)Kipf & Welling (2017), which directly maps node features to binary labels indicating path membership (on the path or not):

$$L_1 : X \to R \tag{1}$$

where X is the stimulus (input features of nodes) and R is the response (binary classification indicating path nodes). Specifically, the GCN computes:

$$Y = \text{GCN}(X, E; \theta) \tag{2}$$

where X are node features, E is the edge index representing maze connectivity, and θ are the GCN parameters. The GCN is trained using gradient descent to minimize a Binary Cross-Entropy (BCE) loss.

The second-order learner is implemented as an MLP-based controller, responsible for dynamically adapting the parameters of the GCN. Given a trained GCN and a novel maze scenario, the MLP controller learns to modify GCN parameters to achieve improved generalization and performance on structurally unknown tasks. Formally, this second-order adaptation can be expressed as:

$$L_2 : (Y, \theta) \to \theta' \tag{3}$$

The adaptation mechanism takes two types of input: (1) the current outputs of the GCN S and (2) the current parameter set θ. The MLP computes parameter updates as follows:

$$\Delta\theta = \text{MLP}(\text{flatten}(Y), \text{flatten}(\theta)) \tag{4}$$

Adapted parameters θ' are then derived as:

$$\theta' = \theta + \Delta\theta \tag{5}$$

The second-order learning itself is performed via gradient-based optimization of the MLP, where the MLP's parameters ϕ are trained to minimize a BCE loss computed using adapted GCN predictions:

$$\mathcal{L}_{\text{MLP}}(\phi) = -\frac{1}{M}\sum_{j=1}^{M}\sum_{i=1}^{N_j}\left[y_{ij}\log(\hat{y}'_{ij}) + (1-y_{ij})\log(1-\hat{y}'_{ij})\right] \tag{6}$$

where $\hat{y}'_{ij}$ is the adapted prediction for node i in maze task j, and M is the number of maze adaptation tasks used in training.

Hierarchical learning occurs through the sequential interaction of the GCN and the MLP controller. For more details, see Appendix.

3.1 Emergence of Representational Isomorphism

A central aspect of our methodology is verifying the emergence of internal cognitive representations isomorphic to maze geometry, which we quantify by computing correlations between pairwise distances in latent embeddings and actual spatial distances in the maze:

$$r_p = \text{Pearson}(D_{\text{latent}}, D_{\text{maze}}), \quad r_s = \text{Spearman}(D_{\text{latent}}, D_{\text{maze}}) \tag{7}$$

High correlation values empirically confirm the representational alignment hypothesis. This can be viewed as representational alignment metric. The Pearson product–moment coefficient measures linear metric fidelity and is most informative when the variables are approximately bivariate normalPearson (1896). Spearman's rank-order coefficient tests for a monotonic relationship, making it robust to non-linear warping and outliersSpearman (1904).

Interpretation. In this setting, a *high* Spearman coefficient (r_s) indicates that the pair-wise ordering of distances in the latent space faithfully mirrors the ordering of true spatial distances i.e., the topology of the maze is preserved. A *high* Pearson coefficient (r_p) further shows that the mapping is approximately linear, meaning metric relations (relative magnitudes, ratios) are maintained as well. Conversely, if r_s is high but r_p is low, the embedding captures the correct qualitative layout (which state is nearer or farther) but distorts exact distances through non-linear warping. If both coefficients are low, the latent geometry fails to align with the physical maze, falsifying the representational-alignment hypothesis. Thus, the pattern of r_s and r_p values jointly diagnoses whether the model has learned a merely ordinal map, a metrically faithful map, or no coherent map at all.

Maze environments are generated algorithmically, creating controlled perturbations to assess adaptation robustness. Each node's feature vector encodes coordinates and structural properties (blockages, degrees), systematically varying across experiments to test adaptation performance rigorously. Our implementation includes delineated algorithmic procedures. See Appendix for a detailed description of the algorithm. The algorithmic setup captures the essence of the theoretical framework by showing that selection for second order learning yields an internal structure (mental representation) that mirrors the external environment. To evaluate the broader utility of second-order learning beyond path classification, we extend our framework to **value-based maze tasks**, where the objective is to predict the **optimal cumulative reward** (value function) for each node. We retain the hierarchical setup: a GCN serves as the first-order learner, while an MLP controller performs second-order adaptation under reward perturbations, operationalizing the theoretical claims of Arnold et al. Arnold et al. (2012; 2015).

The GCN is trained to regress value functions computed via dynamic programming (DP). Given node rewards $R(i,j) = i + j$ and the rightward/downward transition convention used in this experiment, the target value is:

$$V(i,j) = R(i,j) + \max(V(i+1,j), V(i,j+1)) \tag{8}$$

The GCN is optimized using MSE:

$$\mathcal{L}_{\text{GCN}} = \frac{1}{N}\sum_{i=1}^{N}(V_i - \hat{V}_i)^2 \tag{9}$$

Table 1: Large-scale performance on perturbed mazes (mean ± std across seeds).

Model	Accuracy (%)	BCE Loss
Unadapted GCN	62.54 ± 4.63	1.148 ± 0.244
Adapted GCN (SOL)	$\mathbf{80.18 \pm 0.03}$	$\mathbf{0.595 \pm 0.017}$

To evaluate generalization, we introduce **reward perturbations** via random multiplicative masks $\in \{+1, -1\}$. The MLP controller takes as input the predicted values $\hat{V}$ and flattened GCN parameters θ, and predicts parameter updates:

$$\Delta\theta = \text{MLP}(\text{flatten}(\hat{V}), \text{flatten}(\theta)) \tag{10}$$

The adapted parameters are $\theta' = \theta + \Delta\theta$. The adapted GCN is evaluated against DP targets using MSE, and gradients are used to optimize the MLP.

See Appendix for evaluation details

4 Experiments and Results

We empirically validate the efficacy of our hierarchical learning framework, composed of a first-order Graph Convolutional Network (GCN) and a second-order MLP Controller learning mechanisms, across a range of maze navigation and adaptation tasks. Our results demonstrate that second-order learning significantly improves generalization and internal representation structure under distributional shift.

4.1 Experiment 1: Adaptation via MLP Controller

We evaluate a second-order learner that adapts a GCN trained on unperturbed mazes to structurally altered environments.

Setup. We train a base GCN to predict shortest-path node labels on a 10×10 grid maze. Test mazes are generated by randomly removing edges with block probability $p \in \{0.1, 0.2, 0.3, 0.4, 0.5\}$, introducing controlled structural perturbations. An MLP controller takes the GCN outputs and parameters and predicts weight updates for adaptation. Full details of the environment, model architecture, and evaluation protocol are provided in the Appendix.

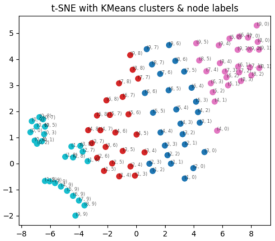
Small-scale evaluation. We first evaluate the model on a small set of unseen mazes to obtain an intuitive understanding of adaptation behavior. The Adapted GCN achieves 93.6% accuracy compared to 61% for the Unadapted GCN, demonstrating effective adaptation. This setting is used for qualitative analysis of adaptation behavior and latent representations.

Large-scale evaluation. To validate these findings statistically, we evaluate across multiple random seeds and a large set of unseen mazes. The Adapted GCN consistently outperforms the Unadapted GCN (Table 1). Accuracy improves from $62.54\% \pm 4.63\%$ to $80.18\% \pm 0.03\%$ ($p = 0.00106$), while BCE loss decreases from 1.148 ± 0.244 to 0.595 ± 0.017 ($p = 0.00344$), indicating robust generalization.

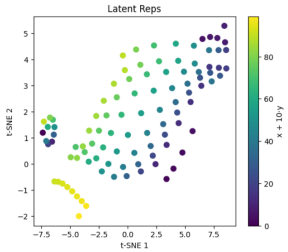
Robustness. Under increasing perturbation, the Unadapted GCN degrades substantially ($0.715 \rightarrow 0.563$), while the Adapted GCN remains stable ($0.810 \rightarrow 0.788$), demonstrating robustness to structural perturbations.

Experimental Scope. Small-scale experiments are used for qualitative analysis of adaptation behavior and latent representations, while large-scale experiments provide statistically robust evaluation of performance.

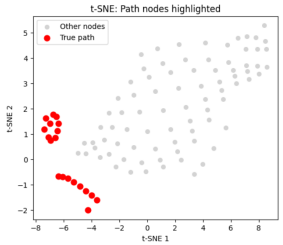
These results show that second-order learning improves both generalization and robustness under distributional shift. We next examine whether this improvement is associated with structured latent representations.



(a) t-SNE projection of latent node embeddings with KMeans cluster assignments (colors) and original maze coordinates (labels).



(b) Latent space colored by spatial coordinate index ($x + 10y$). Clear gradients suggest spatial structure is preserved in the learned representations.



(c) t-SNE plot showing optimal path nodes (red) versus other nodes (gray). Separation indicates meaningful internal abstraction of path information.

Figure 1: Alignment between representations and the environment.

4.2 Latent Representation Visualization:

The following results are obtained from a small set of unseen mazes and are used for qualitative analysis of latent representations. To examine the internal representations learned by the model, we apply t-SNE to the node embeddings from the first GCN layer. t-SNE projections reveal structured latent representations that preserve spatial and functional relationships in the maze (see Appendix for details).

Latent Representation Learning and Functional Clustering We first ask whether the GCN internalises a notion of *path–relevance*. After training the network to label every node in a 10×10 maze as either on– or off–the shortest path from the start $(0, 0)$ to the goal $(9, 9)$, we extract the latent vectors produced by the first GCN layer.[1] We then project the d-dimensional embeddings onto two dimensions with t-SNE for visual inspection. Clustering the same embeddings with KMeans ($k = 4$, selected via the elbow criterion). Figure 1a shows a 2D t-SNE projection of node embeddings of experiment 1, where colors indicate KMeans cluster assignments and node labels reflect original maze coordinates. The clusters are clearly formed and spatially organized, indicating that the GCN develops distinct internal representations for different node roles.

Global Structure of the Latent Space Figure 1a shows the t-SNE projection with KMeans cluster assignments (color) and original maze coordinates (labels). The clusters form a structured layout aligned with the grid topology, indicating that the network encodes both spatial position and functional role. Nearby coordinates remain close in the embedding, while nodes with different roles occupy distinct regions.

Structural Consistency in Latent Representations To evaluate how the latent space reflects the maze structure, we visualize node embeddings from the first GCN layer by coloring points using their spatial index $(x + 10y)$ (Figure 1b). The near-monotonic gradient indicates strong isomorphism: distances in the embedding correlate with distances in the maze. Nodes with similar spatial positions cluster together, showing that the GCN captures meaningful topological structure. Thus, even without coordinate supervision, the model learns an internal map-like representation, supporting effective second-order adaptation across perturbed environments.

Distinct Latent Region for the Optimal Path Figure 1c shows the latent positions of optimal-path nodes (red) versus other nodes (gray). Their tight clustering indicates that the GCN distinctly encodes path-relevant nodes, supporting internalized path abstraction. All path nodes fall into **Cluster 3**, showing that structurally similar nodes are embedded in close proximity.

t-SNE and Isomorphism t-SNE projections show that spatially close nodes remain close in the embedding, with optimal-path nodes forming distinct clusters and smooth spatial gradients. This indicates that the model learns a structured internal representation aligned with the environment.

[1]The dimensionality d of these vectors equals the `hidden_dim` hyper-parameter in the model definition; all results reported here are agnostic to its specific value.

Table 2: OOD generalization performance (accuracy reported as mean ± std).

Model	Accuracy (%)	BCE Loss
Vanilla GCN	57.78 ± 8.38	0.965
Fine-tuned GCN	62.30 ± 6.92	0.830
Reptile-GCN	76.38 ± 4.46	0.582
MAML-GCN	78.66 ± 0.95	0.504
SOL Controller-GCN	**79.33 ± 0.67**	0.511

4.3 Spatial Coherence of the Latent Space

t-SNE projections reveal coherent spatial structure: nodes adjacent in the maze remain close in the embedding, while path-relevant nodes form a compact, distinct region. To quantify this, we compute correlations between pairwise distances in the latent space and the maze using embeddings from the first GCN layer. In Experiment 1, Pearson ($r_p = 0.9042$) and Spearman ($r_s = 0.9085$) correlations are both close to 1, indicating strong alignment between latent and spatial distances. These results provide clear evidence that the model constructs an internal *mental map*, preserving the spatial relationships of the environment and demonstrating environment–cognition isomorphism.

Controller Training Dynamics. The MLP controller exhibits stable training behavior, with loss decreasing across epochs (Fig. 3; see Appendix for details).

4.4 Experiment 2: Comparison with Meta-Learning Baselines

We compare the proposed SOL Controller-GCN with baseline adaptation methods under an out-of-distribution (OOD) setting. Models are trained on mazes with lower perturbation ($p \in \{0.1, 0.2, 0.3\}$) and evaluated on harder unseen mazes ($p \in \{0.4, 0.5\}$). Baselines include Vanilla GCN, Fine-tuned GCN, MAML-GCN Finn et al. (2017), and Reptile-GCN Nichol et al. (2018). Gradient-based methods use five adaptation steps, while SOL performs adaptation via a single forward pass.

Results. As shown in Table 2, SOL achieves the highest accuracy ($79.33\% \pm 0.67\%$), slightly outperforming all baselines and performing comparably to MAML-GCN ($78.66\% \pm 0.95\%$). Reptile-GCN achieves $76.38\% \pm 4.46\%$, while Fine-tuned and Vanilla GCNs perform substantially worse. The improvement over the Vanilla GCN is statistically significant ($p = 0.00572$), indicating improved generalization under distributional shift.

Further analysis of adaptation behavior and computational efficiency is provided in the Appendix.

4.5 Experiment 3: Enriched Latent Input for MLP-based GCN Adaptation

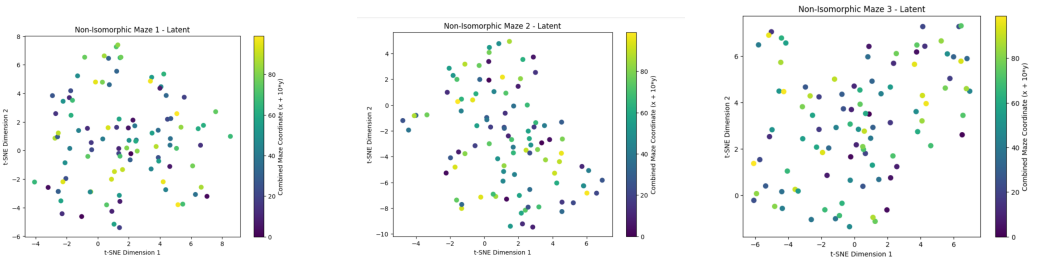
We investigate whether incorporating internal representations improves MLP-based adaptation. The controller receives a concatenation of flattened first-layer activations, output logits, and GCN parameters, yielding an input of:

$$\text{Input Dim} = (\text{hidden_dim} \cdot \text{num_nodes}) + (\text{output_dim} \cdot \text{num_nodes}) + \text{num_params} \tag{11}$$

Results. The enriched controller achieves 93.6% accuracy with a BCE loss of 0.2335 on unseen mazes. While accuracy remains comparable to the baseline, the lower BCE loss indicates improved predictive performance.

4.6 Experiment 4: Generalization Across Maze Sizes

In this experiment, we examine the ability of GCNs trained on smaller mazes to generalize spatial representations to larger maze sizes. The training process used coordinate features (row, column). The Pearson correlation between Euclidean distances in the GCN's latent space and the true grid distances was used as a proxy for spatial structure preservation. This correlation was computed after the first GCN layer, following the ReLU activation, providing a direct measure of how well the learned representation captures the underlying maze geometry. We perform two types of tests.



(a) t-SNE plot of Non-Isomorphic Maze 1 (no spatial alignment).

(b) t-SNE plot of Non-Isomorphic Maze 2 (no spatial alignment).

(c) t-SNE plot of Non-Isomorphic Maze 3 (no spatial alignment).

Figure 2: t-SNE visualizations for three non-isomorphic mazes, all showing a lack of spatial alignment in the learned latent space.

- **Empirical study 1:** The GCN is trained on 8×8 mazes and evaluated directly on larger, unblocked mazes without any additional size-based training. It achieves:
  - 10×10 maze: Pearson correlation = **0.9462**
  - 12×12 maze: Pearson correlation = **0.9559**
  - 14×14 maze: Pearson correlation = **0.9623**
- **Empirical study 2:** The model is trained on both 8×8 and 10×10 mazes to incorporate more size variation. When evaluated on larger sizes:
  - 12×12 maze: Pearson correlation = **0.9570**
  - 14×14 maze: Pearson correlation = **0.9631**

These results show that GCN embeddings generalize effectively to larger mazes, even when trained on smaller ones. High correlation values indicate that the latent space preserves spatial geometry, with slight improvements from increased training diversity. The consistent trend across sizes suggests that the model learns general spatial principles rather than overfitting. This supports the view that GCNs develop transferable, spatially structured representations, enabling second-order adaptation while maintaining core geometric structure and reflecting emergent isomorphic cognitive maps.

4.7 Experiment 5: Representation Collapse in Non-Isomorphic Mazes

We examine the role of isomorphic feature-geometry mapping in enabling internal structure. We replace spatially meaningful node features with i.i.d. Gaussian noise while maintaining the same graph structure. Although the mazes are visually and structurally identical (same connectivity), they no longer provide spatial signals through features. This isolates the impact of representational continuity on GCN performance.

Performance comparison: isomorphic vs. non-isomorphic We evaluate the robustness of the Adapted GCN by comparing performance on isomorphic and non-isomorphic mazes (three each). Accuracy drops from $\sim 91\%$ on isomorphic mazes to $\sim 76\%$ on non-isomorphic mazes. Loss increases sharply from ~ 0.2 to ~ 1.85 in non-isomorphic settings, indicating degradation in the absence of geometric feature cues. Spatial alignment also collapses: Pearson correlation decreases from ~ 0.86 to ~ 0.05, showing a loss of latent structure. To visualize the disruption in learned representations, t-SNE plots are shown for three non-isomorphic mazes in Figure 2. In contrast to isomorphic environments, where the latent space reflected spatial structure, these maps appear randomly distributed.

These findings confirm that meaningful feature-geometry correspondence is critical for effective adaptation and representation formation.

Quantitative Results. Table 3 reports performance on individual non-isomorphic mazes.

Performance deteriorates on non-isomorphic tasks, indicating that the MLP controller benefits from meaningful environment representation alignment.

Table 3: Performance on non-isomorphic mazes.

Maze	Pearson Corr	Adapted (Acc / Loss)	Unadapted (Acc / Loss)
Maze 1	-0.0248	0.78 / 1.8142	0.72 / 0.5588
Maze 2	-0.0580	0.70 / 3.0707	0.55 / 0.6785
Maze 3	-0.0579	0.78 / 0.7160	0.54 / 0.7216

Table 4: Real-world OSM graph adaptation performance (mean $\pm$ std).

Model	Accuracy (%)	BCE Loss	F1
Vanilla GCN	16.53 ± 9.43	5.16 ± 0.70	0.15
Reptile-GCN	53.14 ± 8.20	1.32 ± 0.24	0.19
MAML-GCN	63.25 ± 5.58	$\mathbf{1.11 \pm 0.47}$	**0.23**
SOL Controller-GCN	$\mathbf{75.72 \pm 4.93}$	1.25 ± 0.25	0.19

4.8 Experiment 6: Real-World Graph Adaptation (OSM Road Networks)

We evaluate the proposed method on a real-world road network extracted from OpenStreetMap (Piedmont, CA)Boeing (2025), where road closures simulate distributional shift. We compare SOL Controller-GCN with Vanilla GCN, MAML-GCN, and Reptile-GCN.

Results. As shown in Table 4, SOL achieves the highest accuracy ($75.72\% \pm 4.93\%$), outperforming all baselines and exceeding MAML by over 12% absolute accuracy. Loss remains competitive while maintaining substantially higher accuracy, demonstrating effective adaptation in real-world road-network setting.

Further analysis of metrics and computational efficiency is provided in the Appendix.

4.9 Experiment 7: Second-Order Learning for Value Adaptation

We evaluate second-order learning in a value prediction task under dynamic reward perturbations.

Procedure. A GCN is trained to predict cumulative rewards, and an MLP controller adapts it under reward changes.

Results. Under reward perturbation, adaptation substantially improves performance: MSE decreases from 297.47 to 33.50, MAE from 14.73 to 4.99, and R^2 increases from -2.16 to 0.64. Policy accuracy improves from 25.0% to 66.7%.

These results demonstrate that second-order learning substantially improves value prediction and policy alignment. Additional analysis, reward formulation details, and value-function generation procedures are provided in the Appendix.

5 Conclusion

We propose a hierarchical framework in which a first-order GCN learns task structure while a second-order MLP controller adapts it under environmental changes. Across structural perturbations, OOD transfer, a real-world road network, and value prediction, the controller improves adaptation performance, while the representation analyses show strong environment–latent alignment that degrades under non-isomorphic conditions. Together, these results support an association between effective second-order adaptation and structured, isomorphic first-order representations.

Future work will address **limitations** by expanding to more domains, tasks, and first- and second-order learning algorithms. **Broader impacts** include insights into forces that may drive isomorphism between mental representations and environments in humans.

AI USE STATEMENT

Generative AI tools were used solely for language editing, grammar correction, and stylistic refinement. All AI-assisted edits were reviewed by the authors, who take full responsibility for the final content of the paper.

ETHICS STATEMENT

This work does not involve human subjects, personally identifiable information, or sensitive personal data. The experiments evaluate the proposed framework on controlled maze benchmarks and on a real-world road-network graph derived from publicly available OpenStreetMap data. The proposed second-order learning framework is studied as a general adaptation mechanism for graph-based learning tasks and is not deployed in safety-critical or high-stakes decision-making settings. As with other adaptive machine-learning methods, applications in such settings could introduce risks associated with prediction errors or distribution shift. Accordingly, the results reported here should be interpreted within the scope of the evaluated tasks and environments. We provide experimental, implementation, and reproducibility details to support transparent assessment of the work.

REPRODUCIBILITY STATEMENT

We provide the information needed to reproduce our experiments across the main paper, appendix, and anonymous supplementary materials. The main paper describes the SOL framework, experimental settings, baselines, and evaluation procedures. Additional implementation and experimental details, including model architecture, training procedures, task construction, hyperparameters, random seeds, evaluation metrics, and statistical analyses, are provided in the appendix. The appendix also maps the reported experiments to the corresponding implementation notebooks. The anonymous supplementary materials include the code and notebooks used for the synthetic maze experiments, large-scale evaluations, meta-learning comparisons, representation analyses, OpenStreetMap road-network experiments, and value-adaptation experiments.

The anonymous supplementary repository is available at:

`https://anonymous.4open.science/r/SOL_ICLR2027_GitHub_Repo-05A8/`

REFERENCES

Hervé Abdi and Lynne J Williams. Principal component analysis. *Wiley interdisciplinary reviews: computational statistics*, 2(4):433–459, 2010.

Solvi Arnold, Reiji Suzuki, and Takaya Arita. Second order learning and the evolution of mental representation. In *Artificial Life Conference Proceedings*, pp. 301–308. MIT Press, 2012.

Solvi Arnold, Reiji Suzuki, and Takaya Arita. Selection for representation in higher-order adaptation. *Minds and Machines*, 25:73–95, 2015.

Andrea Banino, Caswell Barry, Benigno Uria, Charles Blundell, Timothy Lillicrap, Piotr Mirowski, Alexander Pritzel, Martin J Chadwick, Thomas Degris, Joseph Modayil, et al. Vector-based navigation using grid-like representations in artificial agents. *Nature*, 557(7705):429–433, 2018. doi: 10.1038/s41586-018-0102-6.

Peter W Battaglia, Jessica B Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, et al. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*, 2018.

Geoff Boeing. Modeling and analyzing urban networks and amenities with osmnx. *Geographical Analysis*, 57(4):567–577, 2025.

Christopher J Cueva and Xue-Xin Wei. Emergence of grid-like representations by training recurrent neural networks to perform spatial localization. *arXiv preprint arXiv:1803.07770*, 2018. URL `https://arxiv.org/abs/1803.07770`.

Xu Dehong, Gao Ruiqi, Zhang Wen-Hao, Wei Xue-Xin, and Wu Ying Nian. On conformal isometry of grid cells: Learning distance-preserving position embedding. In *The Thirteenth International Conference on Learning Representations*, 2025.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. *Proceedings of the 34th International Conference on Machine Learning*, pp. 1126–1135, 2017.

David Ha, Andrew M. Dai, and Quoc V. Le. Hypernetworks. In *Proceedings of the 5th International Conference on Learning Representations (ICLR)*, 2017. URL https://arxiv.org/abs/1609.09106.

Harry F. Harlow. The formation of learning sets. *Psychological Review*, 56(1):51–65, 1949. doi: 10.1037/h0062474.

Bo Jiang, Yujun Zhang, Jian Wang, Yiqiang Wang, and Wei Zhang. Meta-learning graph neural networks for few-shot node classification. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management (CIKM)*, pp. 2697–2705, 2019. doi: 10.1145/3357384.3358136.

Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *International Conference on Learning Representations (ICLR)*, 2017.

Edvard I Moser, Emilio Kropff, and May-Britt Moser. Place cells, grid cells, and the brain's spatial representation system. *Annual Review of Neuroscience*, 31:69–89, 2008.

Andreas Müller, Carlo Curino, and Raghu Ramakrishnan. Mothernet: Fast training and inference via hyper-network transformers. In *International Conference on Learning Representations*, volume 2025, pp. 76666–76686, 2025.

Alex Nichol, Joshua Achiam, and John Schulman. On first-order meta-learning algorithms. *arXiv preprint arXiv:1803.02999*, 2018. URL https://arxiv.org/abs/1803.02999.

Karl Pearson. Mathematical contributions to the theory of evolution. III. regression, heredity, and panmixia. *Philosophical Transactions of the Royal Society of London A*, pp. 253–318, 1896.

Charles Spearman. The proof and measurement of association between two things. *American Journal of Psychology*, 15(1):72–101, 1904.

Kenneth O. Stanley, David B. D'Ambrosio, and Jason Gauci. A hypercube-based encoding for evolving large-scale neural networks. *Artificial Life*, 15(2):185–212, 2009. doi: 10.1162/artl.2009.15.2.15202.

Edward C Tolman. Cognitive maps in rats and men. *Psychological Review*, 55(4):189–208, 1948.

Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

Erik Wijmans, Manolis Savva, Irfan Essa, Stefan Lee, Ari S Morcos, and Dhruv Batra. Emergence of maps in the memories of blind navigation agents. *AI Matters*, 9(2):8–14, 2023.

Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S Yu. A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 32(1):4–24, 2021. doi: 10.1109/TNNLS.2020.2978386.

Daniel Zügner and Stephan Günnemann. Adversarial attacks on graph neural networks via meta learning. In *Proceedings of the 7th International Conference on Learning Representations (ICLR)*, 2019. URL https://arxiv.org/abs/1902.08412.

A Experimental Setup Details

A.1 Maze Environment

We construct environments as 10×10 grid graphs using `networkx`. Each node corresponds to a coordinate (i, j) and is connected to its four neighbors, except at boundaries. The start node is fixed at $(0, 0)$ and the goal at $(9, 9)$.

To simulate structural perturbations, edges are randomly removed with probability $p \in \{0.1, 0.2, 0.3, 0.4, 0.5\}$. For large-scale experiments, mazes are regenerated until a valid path exists between start and goal.

A.2 Node Features

Each node is represented by a 5-dimensional feature vector:

$$[x, y, \text{blockage count}, \text{original degree}, \text{current degree}]$$

which encodes both spatial position and local structural changes.

A.3 Task Definition

The task is binary node classification: predicting whether a node lies on the shortest path between start and goal. Ground-truth labels are computed using shortest-path algorithms.

A.4 Model Architecture

The first-order learner is a two-layer GCN:

- Input dimension: 5
- Hidden dimension: 16
- Output dimension: 1

The second-order learner is an MLP controller that predicts parameter updates for the GCN. Its input consists of flattened latent representations, GCN outputs, and current GCN parameters. The MLP has one hidden layer (64 units), LeakyReLU activation, and dropout (0.5).

A.5 Training Procedure

The base GCN is trained on an unperturbed maze ($p = 0$) using binary cross-entropy loss and Adam (learning rate 0.01).

The MLP controller is trained across multiple perturbed mazes. For the residual-update implementations, parameter updates $\Delta\theta$ are applied as:

$$\theta' = \theta + \Delta\theta$$

The adapted parameters are used via functional forward passes, while GCN parameters remain frozen. Implementation-specific variants are described in the Additional Experimental Details.

A.6 Evaluation Protocol

Small-scale:

- Training tasks: 5
- Test tasks: 3

Used for qualitative analysis and representation study.

Large-scale:

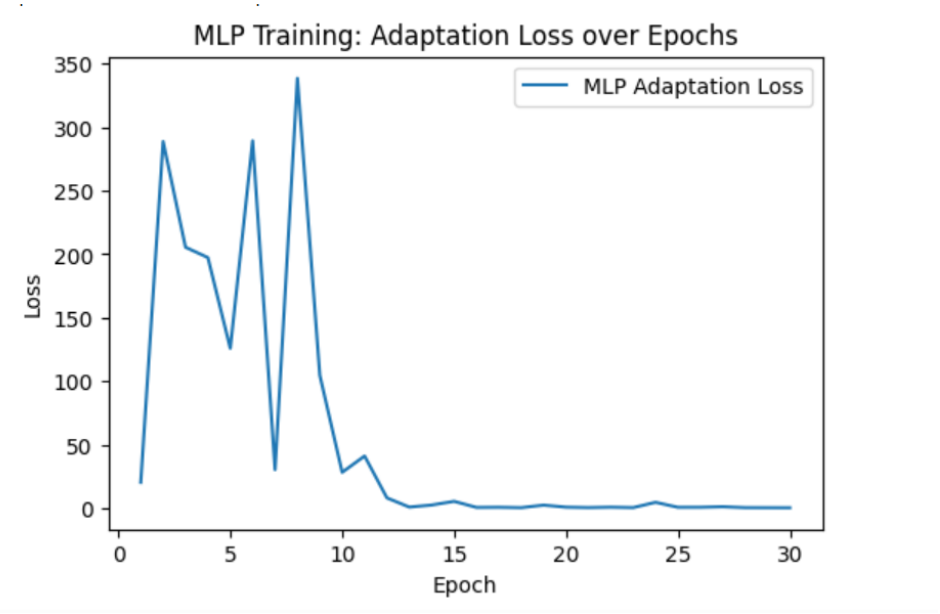


Figure 3: Training loss of the MLP Controller across epochs.

- Seeds: 5
- Training tasks per seed: 50
- Test tasks per seed: 200

Results are aggregated as mean $\pm$ standard deviation across seeds.

A.7 t-SNE Visualization Details

To analyze the geometry of the learned representations, we use t-SNE, a non-linear dimensionality reduction method that preserves local neighborhood structure by minimizing Kullback–Leibler divergence between high- and low-dimensional similarities Van der Maaten & Hinton (2008).

This makes it effective for visualizing latent structure and revealing clustering patterns. In contrast, linear methods such as PCA are less suitable for capturing non-linear relationships Abdi & Williams (2010).

A.8 Controller Training Dynamics

The MLP's loss exhibits substantial early fluctuations but decreases sharply after the initial epochs and stabilizes at a low value (Fig. 3), indicating convergence of controller training.

Across blocked mazes, the adapted GCN achieves lower MSE and higher policy accuracy than the unadapted baseline, indicating that the learned updates are functionally meaningful.

The controller also generalizes across tasks, block rates, and maze layouts, suggesting a transferable inductive bias.

B Additional Experimental Details

B.1 Algorithmic Description

Below, we detail the main algorithms in pseudocode.

Algorithm 1 Create Maze Graph

Input: Maze size n, block probability p
$G \leftarrow$ 2D grid graph of size $n \times n$
original_edges $\leftarrow E(G)$
original_degrees $\leftarrow \{\deg(v) \mid v \in V(G)\}$
edges $\leftarrow$ List$(E(G))$
Shuffle edges
num_blockages $\leftarrow \lfloor p \cdot |\text{edges}| \rfloor$
removed_edges $\leftarrow \varnothing$
for edge in first num_blockages edges **do**
    Remove **edge** from G
    Append **edge** to removed_edges
end for
For each node v in G:
    Compute feature vector $f(v) = [x(v), y(v),$
    # blockages at v, original degree, current degree]
node_features $\leftarrow \{v \mapsto f(v)\}$
Output: G, edge_index, node_features, original_edges, removed_edges, start, goal

Algorithm 2 Train GCN on Original Maze

Input: Maze graph G, node features, edge index, labels derived from the maze shortest path.
Initialize GCN with parameters θ.
for epoch $= 1$ to T **do**
    Forward propagate:
        $\hat{y} \leftarrow \text{GCN}(\text{node_features}, \text{edge_index};\ \theta)$
    Compute loss:
        $\mathcal{L} \leftarrow \text{BCE}(\hat{y}, y)$
    Backpropagate and update parameters:
        $\theta \leftarrow \theta - \eta \nabla_\theta \mathcal{L}$
end for
Visualize latent representations using t-SNE.
Compute distance correlations between latent space and maze coordinates.
Output: Trained GCN.

Algorithm 3 Train MLP Controller for GCN Adaptation

Input: Set of blocked maze tasks $\{G_i\}$, pretrained GCN with parameters θ, number of epochs T_2.
for epoch $= 1$ to T_2 **do**
    for each maze task i **do**
        Compute GCN output on task i:
            $\hat{y}_i \leftarrow \text{GCN}(\text{features}_i, \text{edge_index}_i;\ \theta)$
        Flatten $\hat{y}_i$ and current GCN weights θ into a single vector.
        MLP Controller outputs adapted parameters:
            $\theta_i' \leftarrow \text{MLP}(\hat{y}_i, \theta)$
        Compute adapted output:
            $\hat{y}_i' \leftarrow \text{GCN}(\text{features}_i, \text{edge_index}_i;\ \theta_i')$
        Compute loss: $\mathcal{L}_i \leftarrow \text{BCE}(\hat{y}_i', y_i)$
        Backpropagate and update MLP parameters.
    end for
    Record average loss:
        $\mathcal{L}_{\text{total}} = \frac{1}{N} \sum_i \mathcal{L}_i$
end for
Output: Trained MLP Controller.

B.2 Additional Analysis for Experiment 2

Loss Comparison. The SOL Controller-GCN achieves a BCE loss of 0.511, comparable to MAML-GCN (0.504) and lower than Reptile-GCN (0.582) and Fine-tuned GCN (0.830). The reduction compared to the Vanilla GCN (0.965) is statistically significant ($p = 0.01827$).

Algorithm 4 Evaluation and Distance Correlation Computation

Input: Unseen maze task G, features, edge index.
Obtain latent representations:
    $Z \leftarrow \text{ReLU}(\text{GCN.conv1}(\text{node_features}, \text{edge_index}))$
Compute pairwise distances D_Z in the latent space.
Extract maze coordinates M for each node and compute pairwise distances D_M.
Flatten the upper triangles of D_Z and D_M to vectors z and m.
Compute Pearson correlation r_p and Spearman correlation r_s between z and m.
Output: r_p, r_s (high values indicate strong preservation of maze geometry in Z).

Computational Efficiency. SOL performs adaptation via a single forward pass, requiring approximately 0.014 seconds. In contrast, gradient-based methods require multiple forward and backward passes (five steps), resulting in 0.047–0.050 seconds. This corresponds to a $\sim 3.5\times$ speedup.

Adaptation Mechanism. Unlike gradient-based methods (Fine-tuning, MAML, Reptile), which require test-time gradients and labeled data, SOL predicts parameter updates directly from the model state in a single forward pass. This enables efficient, label-free adaptation.

Interpretation. These results show that SOL achieves performance comparable to MAML while significantly reducing adaptation cost. The learned controller captures a transferable adaptation strategy, eliminating the need for explicit optimization at test time.

B.3 Additional Analysis for Experiment 6

Setup. We construct a road network graph from OpenStreetMap (Piedmont, CA) with 250 nodes and 322 edges, where nodes represent intersections and edges represent road segments. A base GCN is trained to predict shortest-path nodes, and road-closure tasks are generated by randomly removing edges while preserving connectivity.

Additional Metrics. While MAML achieves slightly higher F1 and IoU, SOL remains competitive and attains higher overall accuracy. This reflects different prediction behavior: MAML achieves higher recall by predicting larger path sets, whereas SOL produces more conservative predictions with fewer false positives.

Computational Efficiency. SOL performs adaptation via a single forward pass ($\sim$ 0.0077s), whereas MAML and Reptile require multiple gradient steps ($\sim$ 0.018–0.020s), yielding approximately $2.5\times$ faster adaptation.

Adaptation Mechanism. SOL performs gradient-free, label-free adaptation at test time, unlike gradient-based meta-learning methods that require iterative optimization and labeled data.

Interpretation. These results demonstrate that second-order learning generalizes to real-world graph structures, achieving high accuracy with efficient and transferable adaptation.

B.4 Additional Analysis for Experiment 7

These results provide evidence for the effectiveness of second-order learning in dynamic environments, enabling rapid adaptation and improved representations.

In this experiment, the base GCN functions as a first-order learner that approximates the value function over the maze graph, while the MLP controller acts as a second-order learner that dynamically adapts the GCN parameters under environmental perturbations. The hierarchical interaction between these two learning processes enables the system to maintain coherent value estimates even when the reward structure changes significantly.

From a hierarchical learning perspective, adaptation at the second-order level (MLP controller) induces structure at the first-order level (GCN representations). Rather than memorizing individual maze configurations, the controller learns transferable parameter transformations that generalize across reward perturbations and blocked environments. This aligns with theoretical perspectives suggesting that second-order learning promotes structured and adaptive representations.

The substantial improvement in MSE, MAE, and R^2 after adaptation indicates that the second-order learner successfully restructures the value representation to match altered reward landscapes. Furthermore, the increase in policy accuracy demonstrates that the adapted value estimates are behaviorally meaningful, producing action selections that more closely align with optimal trajectories obtained via dynamic programming.

We hypothesize that effective second-order adaptation relies on isomorphic alignment between learned representations and the environment. The GCN develops latent representations that preserve important relational properties of the maze, while the MLP controller exploits this structured geometry to efficiently adapt the network under changing conditions. Such alignment enables the system to learn how to adapt efficiently under changing conditions and supports the emergence of learning-to-learn behavior.

More broadly, these findings support the view that higher-order optimization processes exert organizational pressure on lower-level representations. Selection pressure at the adaptation level therefore encourages the emergence of structured internal representations that facilitate robust generalization and flexible behavior in dynamic environments. The improved adaptation performance suggests that the learned value representations preserve structural information about the environment, extending the representational isomorphism hypothesis beyond shortest-path classification tasks.

Reward Calculation and Value Function Generation. The environment is represented as a two-dimensional grid graph $G = (V, E)$ of size $n \times n$, where each node corresponds to a spatial location (i, j) in the maze and edges connect neighboring grid positions. The start node is fixed at the top-left corner $(0, 0)$, while the goal node is fixed at the bottom-right corner $(n - 1, n - 1)$.

For the value-adaptation experiment, each maze node (i, j) is assigned a deterministic reward value defined as:

$$R(i, j) = i + j \tag{12}$$

This reward structure creates a monotonic reward gradient across the maze, where rewards increase as the agent moves closer to the goal state. The resulting environment encourages trajectories that maximize cumulative future reward while preserving the spatial structure of the grid.

Ground-truth state values are computed using dynamic programming (DP). Starting from the goal state, the optimal cumulative reward for each node is recursively calculated as:

$$V(i, j) = R(i, j) + \max(V(i + 1, j), V(i, j + 1)) \tag{13}$$

where the recursion considers valid rightward and downward transitions in the maze. The terminal goal node is initialized with:

$$V(n - 1, n - 1) = R(n - 1, n - 1) \tag{14}$$

To create dynamic reward environments, perturbations are introduced through multiplicative reward masks sampled from $\{+1, -1\}$. A subset of maze locations is therefore assigned negative rewards, simulating blocked or penalized regions:

$$R'(i, j) = R(i, j) \cdot m(i, j), \quad m(i, j) \in \{+1, -1\} \tag{15}$$

The probability of assigning a negative reward is controlled by a blockage parameter $p \in \{0.1, 0.2, 0.3\}$. These perturbations alter the optimal value landscape and require the second-order learner to dynamically adapt the GCN parameters in order to maintain accurate value estimation and policy behavior.

B.5 Evaluation metrics

Path–prediction experiment (shortest–path classification)

- **Binary-cross-entropy (BCE) loss & classification accuracy:** measured on unseen blocked mazes to quantify generalisation of the path/non-path discriminator.
- **Representational isomorphism:** Pearson and Spearman correlations between (i) pairwise Euclidean distances in the first-layer latent space and (ii) pairwise Manhattan distances in the maze layout. High positive values indicate that the internal geometry mirrors the external environment.

Value-adaptation experiment (dynamic reward map).

- **Mean-squared error (MSE)** and **mean-absolute error (MAE)** between predicted and ground-truth state-value functions.
- **Coefficient of determination (R^2):** proportion of variance in the true values explained by the model.
- **Policy accuracy:** fraction of nodes at which the adapted model's recommended action matches the optimal action obtained via dynamic programming.

These metrics capture both regression accuracy and policy-level behavior, providing a comprehensive evaluation of second-order learning performance in dynamic reward environments.

B.6 Hierarchical Learning Procedure

1. **First-order training:** Train the base GCN on an initial set of unperturbed maze navigation tasks to convergence, establishing baseline parameters θ.
2. **Second-order training (adaptation phase):** Given a set of structurally perturbed maze environments (e.g., mazes with randomly blocked paths), the MLP controller learns to dynamically adapt the GCN parameters. Adaptation is performed in a differentiable manner, allowing backpropagation from the GCN outputs through the adapted parameters to the MLP controller.

C Extended Reproducibility and Implementation Notes

The preceding sections provide the core experimental details used throughout the paper. This section adds implementation-level information from the supplied experiment code in order to make the relationship between the reported results and the computational pipeline more explicit.

C.1 End-to-End SOL Computation

The implementation separates first-order prediction from second-order adaptation. Let the first-order GCN be f_θ and the MLP controller be g_ϕ. On a task (X, E), the GCN first produces an internal state and output,

$$H^{(1)} = \mathrm{ReLU}(\mathrm{GCNConv}_1(X, E)), \qquad \hat{Y} = f_\theta(X, E). \tag{16}$$

The controller then maps the current first-order state together with the current GCN parameter vector to a task-conditioned parameter change,

$$\Delta\theta = g_\phi(s_\theta, \theta). \tag{17}$$

In the large-scale, OOD-comparison, and OSM implementations, this update is applied residually,

$$\theta' = \theta + \alpha\Delta\theta. \tag{18}$$

The large-scale maze implementation uses a unit update scale ($\alpha = 1$), whereas the OOD meta-learning comparison and OSM implementation use $\alpha = 0.1$.

During second-order controller training, the base GCN parameters remain fixed while gradients flow through the functional forward pass using θ' in order to optimize ϕ. During SOL test-time adaptation, the controller produces $\Delta\theta$ in a forward computation; no iterative test-time gradient update and no test label are used.

C.2 Controller Input Variants

The experiments use two closely related controller input forms.

Output-and-parameter form. The basic controller implementation receives the flattened GCN outputs and current GCN parameters:

$$s_{\mathrm{basic}} = [\mathrm{vec}(\hat{Y}); \mathrm{vec}(\theta)]. \tag{19}$$

Enriched latent form. The enriched controller input additionally includes the first-layer GCN activations. This form is used in Experiment 3 and in the large-scale, OOD-comparison, and OSM implementations:

$$s_{\text{enriched}} = [\text{vec}(H^{(1)}); \text{vec}(\hat{Y}); \text{vec}(\theta)]. \tag{20}$$

For N nodes, hidden dimension d_h, output dimension d_o, and GCN parameter count $|\theta|$, the enriched controller input dimension is

$$d_{\text{ctrl}} = Nd_h + Nd_o + |\theta|. \tag{21}$$

For the principal 10×10 maze with $N = 100$, $d_h = 16$, and $d_o = 1$, the node-dependent portion contributes 1700 scalar inputs before appending the GCN parameter vector.

C.3 Functional Parameter Application

The adapted parameter vector is partitioned back into tensors matching the original GCN parameter shapes and supplied to a functional forward call. This permits a distinct task-conditioned parameterization to be evaluated without permanently overwriting the reference GCN. The controller therefore learns a transformation in parameter space rather than continuing ordinary test-time gradient training of the first-order learner.

C.4 Random Seed Control

The large-scale and OOD meta-learning implementations seed Python's `random` module, NumPy, PyTorch, and CUDA from the same experiment seed. The principal five-seed evaluations use

$$\{0, 1, 2, 3, 4\}. \tag{22}$$

These seeds affect model initialization and stochastic task generation, including edge-removal patterns.

C.5 Execution Hardware Recorded in the Notebooks

The saved large-scale maze and OSM notebook outputs report CUDA execution on an NVIDIA GeForce RTX 3060 Laptop GPU.

D Extended Experiment 1: Adaptation Robustness and Detailed Breakdown

This section expands the analysis of the primary shortest-path adaptation experiment without changing its task, model, or reported headline result.

D.1 Small-Scale Qualitative Evaluation

The small-scale setting is used to make the adaptation mechanism directly inspectable. The base GCN is trained on the unperturbed 10×10 maze and the controller is trained on five perturbed mazes. Three unseen perturbed mazes are then evaluated.

The main paper reports an average adapted accuracy of 93.6% compared with 61% for the unadapted GCN in the corresponding small-scale comparison. The purpose of this evaluation is qualitative rather than statistical: it supports direct visualization of latent organization, path separation, and the behavior of individual adapted predictions.

D.2 Large-Scale Statistical Evaluation

The large-scale experiment repeats the adaptation procedure over five seeds, with 50 controller-training tasks and 200 unseen test tasks per seed. The headline performance results are reported in the main paper.

The paired tests reported in the main paper give $p = 0.00106$ for accuracy and $p = 0.00344$ for BCE loss. The large-scale evaluation therefore supports the small-scale observation with a repeated multi-seed analysis.

Table 5: Experiment 1 performance broken down by edge-removal probability. The table makes explicit the robustness trend summarized in the main paper.

p	Unadapted Acc.	SOL Acc.	Unadapted BCE	SOL BCE
0.1	0.7153	0.8100	0.9429	0.6029
0.2	0.6537	0.8097	1.0954	0.5921
0.3	0.6260	0.8078	1.0973	0.5854
0.4	0.5815	0.7950	1.2628	0.5986
0.5	0.5632	0.7875	1.3151	0.6002

D.3 Breakdown by Structural Perturbation Strength

The existing large-scale evaluation also records performance separately for each edge-removal probability. Table 5 exposes this breakdown. These values correspond to the same task generator, model, and evaluation pipeline as the aggregate Experiment 1 result.

The effect of structural perturbation is markedly different for the unadapted and SOL-adapted models. As the edge-removal probability increases from $p = 0.1$ to $p = 0.5$, the unadapted GCN accuracy decreases from 0.7153 to 0.5632, corresponding to a reduction of 15.21 percentage points. In contrast, the SOL-adapted GCN decreases only from 0.8100 to 0.7875, a reduction of 2.25 percentage points over the same perturbation range. Consequently, the accuracy separation between the two models increases as the environment becomes more strongly perturbed, reaching 22.43 percentage points at $p = 0.5$.

A corresponding difference is observed in predictive loss. The unadapted BCE increases from 0.9429 at $p = 0.1$ to 1.3151 at $p = 0.5$, an increase of 0.3722. In contrast, the adapted BCE remains concentrated near 0.6 across all five perturbation levels, ranging only from 0.5854 to 0.6029. Thus, while increasingly severe edge removal substantially degrades the frozen first-order model, the task-conditioned parameter transformation produced by the second-order controller maintains comparatively stable predictive performance.

Figure 4 provides a complementary visual summary of this robustness analysis. Panels (a) and (b) plot the existing five-seed mean accuracy and BCE values from Table 5. Panel (c) shows the overall adapted-minus-unadapted accuracy improvement for each of the five large-scale experiment seeds. These paired seed-level gains are 17.51, 12.48, 24.48, 14.44, and 19.31 percentage points, respectively. Their mean improvement is approximately 17.6 percentage points, with a 95% Student-t confidence interval of approximately $[11.9, 23.4]$ percentage points. The associated paired comparison is consistent with the significant overall accuracy improvement reported in the large-scale experiment ($p = 0.00106$).

Taken together, the tabulated and visualized results show that the benefit of second-order adaptation is not restricted to a single perturbation level. Rather, the SOL controller maintains relatively stable accuracy and loss as structural disruption increases, while the unadapted GCN becomes progressively less reliable. This pattern provides a more detailed view of the robustness result summarized in the main paper and supports the interpretation that the controller learns a transferable adaptation strategy across varying degrees of graph perturbation.

D.4 Controller Training Dynamics

The MLP controller is trained across a diverse collection of structurally perturbed maze tasks while the parameters of the first-order GCN remain fixed. For each training task, the GCN produces its current internal representation and prediction, which are supplied to the second-order controller together with the current GCN parameters. The controller then generates a task-conditioned parameter transformation, and the resulting adapted parameterization is evaluated through a functional GCN forward pass. Gradients from the adapted task loss optimize the controller, enabling the second-order learner to acquire a reusable adaptation strategy across perturbations while preserving the reference first-order model.

The controller-loss trajectory shown in Fig. 3 demonstrates stable second-order optimization. The progressive reduction in loss indicates that the controller learns increasingly effective parameter

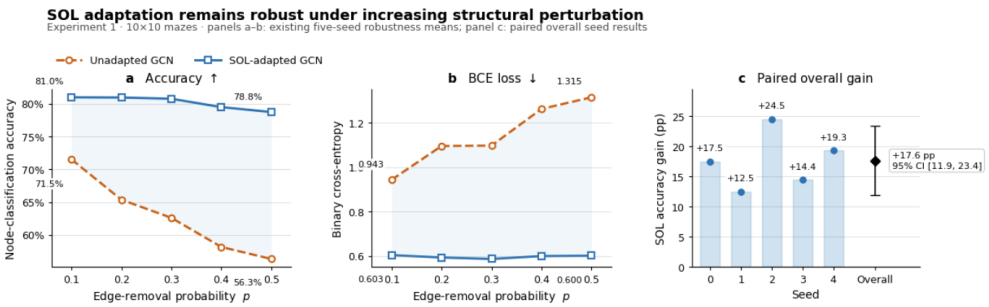


Figure 4: **Robustness of SOL adaptation under increasing structural perturbation.** **(a)** Node-classification accuracy as the edge-removal probability increases from $p = 0.1$ to $p = 0.5$. The unadapted GCN exhibits a pronounced decline as progressively more graph structure is removed, whereas the SOL-adapted GCN remains comparatively stable. **(b)** Binary cross-entropy loss across the same perturbation range. The loss of the unadapted GCN increases substantially with structural disruption, while SOL remains near 0.6 across all tested perturbation levels. **(c)** Overall paired accuracy improvement of SOL relative to the unadapted GCN for each of the five large-scale experiment seeds. Bars and blue markers denote the seed-level paired gains. The black diamond denotes the mean paired improvement, and the error bar denotes its 95% Student-t confidence interval. Panels (a) and (b) therefore describe the perturbation-dependent five-seed means, whereas panel (c) summarizes the overall paired effect across seeds.

Table 6: Training and test-time roles of the second-order controller in Experiment 1.

	Controller training	**Test-time adaptation**
Environment	Perturbed training mazes	Previously unseen perturbed mazes
Reference GCN	Fixed	Fixed
Controller parameters	Learned across training tasks	Fixed after training
Task labels	Used to define the controller-training objective	Not required for adaptation
Gradient computation	Used to learn the controller	Not required at test time
Adaptation mechanism	Learn a transferable mapping from model state to parameter transformation	Apply the learned transformation in a forward computation

transformations across the perturbed training environments. This learning process is reflected in the large-scale evaluation in Sections D.2 and D.3, where the same learned controller is applied directly to previously unseen structural perturbations.

For the large-scale Experiment 1 evaluation, each random seed uses 50 perturbed mazes for controller training and 200 unseen perturbed mazes for evaluation. The controller is optimized for 30 epochs while the first-order GCN remains fixed. Once this second-order learning phase is complete, the controller parameters are held fixed and the learned adaptation rule is reused across all unseen test environments. The controller is therefore not re-optimized for individual test mazes; instead, it directly generates a task-conditioned parameter transformation from the current model state.

This separation between controller learning and controller application is a central property of SOL. During training, the second-order learner acquires a mapping from the state of the first-order model to a task-conditioned parameter transformation. At test time, this learned mapping is applied directly, eliminating the need for an iterative inner optimization loop, test-time gradients, or labeled test examples. In this sense, SOL amortizes the adaptation process: experience gathered across per-

turbed training environments is encoded into a reusable second-order mechanism that can rapidly configure the first-order learner for a new environment.

The robustness results in Fig. 4 further demonstrate the effectiveness of this learned adaptation mechanism. As edge-removal probability increases, the unadapted GCN exhibits a substantial decline in accuracy and increase in BCE loss, whereas the SOL-adapted model remains comparatively stable across the full perturbation range. Because the same trained controller is applied across independently generated unseen mazes, these results show that the learned second-order mapping transfers across variations in graph connectivity rather than being tied to a specific perturbation instance.

Together, the controller-training trajectory and the large-scale unseen-maze results provide complementary evidence for the proposed second-order learning mechanism. The training dynamics show that the controller learns a stable task-conditioned parameter transformation across perturbed environments, while the held-out evaluations show that this learned transformation can be applied directly to novel structural configurations. The resulting combination of transferable adaptation, single-pass application, and the absence of test-time gradients or labeled test data is a defining advantage of the SOL framework.

E Extended Representational Isomorphism and Latent-Space Analysis

A central component of the SOL framework is the emergence of structured first-order representations that reflect the organization of the environment. The main paper provides qualitative visualizations of this phenomenon together with the principal quantitative isomorphism result. This appendix expands that analysis by describing the representation-extraction procedure, formalizing the distance-based alignment metric, examining the functional and spatial organization of the learned latent space, and reporting the stability of the alignment across independent large-scale runs.

Taken together, these analyses show that the first-order GCN develops a latent geometry that preserves important relational properties of the maze while also organizing nodes according to their task-relevant role. This structured internal state provides an informative substrate on which the second-order controller can operate when adapting the first-order learner to a structurally changed environment.

E.1 Latent Representation Extraction

Representations are extracted from the first GCN layer after the ReLU nonlinearity:

$$Z = \mathrm{ReLU}\left(\mathrm{GCN.conv1}(X, E)\right). \tag{23}$$

For the principal 10×10 maze experiments, $Z \in \mathbb{R}^{100\times 16}$, such that each of the 100 maze nodes is represented by a 16-dimensional latent vector.

The first GCN layer provides a particularly useful representation for this analysis because it lies between the structured graph input and the final scalar path-classification output. Examining this intermediate state therefore allows us to characterize how spatial and task-relevant information is organized internally before it is compressed into the final binary decision.

The node features contain spatial and local structural information, including the node coordinates and graph-degree statistics. The representational analysis asks how these inputs are transformed by task learning: specifically, whether the resulting high-dimensional latent geometry preserves coherent relational structure corresponding to the maze while simultaneously organizing nodes according to their functional role in shortest-path prediction.

E.2 Pairwise-Distance Alignment

To quantify the correspondence between the learned internal representation and the external maze, we compare pairwise distances between nodes in the latent space with pairwise distances between their corresponding spatial locations.

For latent vectors Z_i and Z_j, the representational distance is

$$D_{\text{latent}}(i, j) = \|Z_i - Z_j\|_2. \tag{24}$$

Let $D_{\text{maze}}(i, j)$ denote the corresponding spatial distance between the maze coordinates of nodes i and j. The upper-triangular entries of the two distance matrices are vectorized and compared using Pearson and Spearman correlation:

$$r_p = \text{Pearson}\left(D_{\text{latent}}, D_{\text{maze}}\right), \qquad r_s = \text{Spearman}\left(D_{\text{latent}}, D_{\text{maze}}\right). \tag{25}$$

The two statistics characterize complementary aspects of environment–representation alignment. Pearson correlation measures the linear association between latent and environmental distances, while Spearman correlation measures preservation of their relative ordering.

Consequently, high values for both statistics indicate strong linear and ordinal alignment between distances in the learned latent space and the underlying maze. This pairwise-distance analysis provides the quantitative operationalization of environment–representation isomorphism used throughout the paper.

E.3 Qualitative Visualization of the Learned Geometry

The main paper complements the distance-based analysis with three qualitative visualizations of the first-layer representation in Fig. 1. Rather than reproducing those panels in the appendix, we use this section to provide the corresponding analysis protocol and a more detailed interpretation of what each visualization demonstrates.

For visualization, the 16-dimensional first-layer node representations are projected to two dimensions using t-SNE Van der Maaten & Hinton (2008). The implementation uses `random_state=42` for the principal visualizations.

Importantly, t-SNE is used as a visualization of latent organization, whereas the reported Pearson and Spearman isomorphism measurements are computed from pairwise distances in the original latent representation rather than from the two-dimensional t-SNE projection. Thus, the qualitative and quantitative analyses provide complementary views of the same learned representation.

The three panels in Fig. 1 examine the representation from three perspectives: cluster organization, spatial organization, and task-relevant path organization. Together they reveal that the first-order latent space contains structure associated with both the geometry of the maze and the functional requirements of shortest-path prediction.

E.4 Functional Organization of the Latent Space

The first qualitative analysis examines whether the internal representation organizes nodes according to functionally meaningful roles. After training the GCN for shortest-path node classification, the first-layer representations are projected to two dimensions using t-SNE, and the resulting projection is analyzed with KMeans clustering using $k = 4$. As shown in the cluster visualization in Fig. 1, the latent representations form organized regions rather than an arbitrary intermixing of node embeddings. The cluster structure is accompanied by systematic variation in the maze coordinates associated with the embedded nodes, indicating that the learned representation retains structured information about the graph environment.

This organization is particularly relevant for SOL because the first-order representation is not merely a passive encoding of individual node features. It is produced after graph convolution has combined node information with neighborhood structure in service of the shortest-path prediction objective. The resulting latent state therefore provides the second-order learner with a structured description of the current first-order computation.

E.5 Spatial Organization of the Latent Representation

A second visualization in Fig. 1 colors the latent representations according to the spatial index

$$s(x, y) = x + 10y. \tag{26}$$

The resulting t-SNE projection exhibits a smooth spatial progression. Nodes associated with nearby maze locations tend to occupy nearby regions of the projection, while nodes corresponding to progressively different locations form an ordered transition through the latent space.

This observation is consistent with the pairwise-distance results reported below. The learned GCN does not simply retain node coordinates independently; through graph convolution and task training, spatial and structural input information is transformed into a distributed 16-dimensional representation. The observed gradient shows that this learned representation maintains a coherent organization corresponding to the relational geometry of the environment.

The combination of local graph processing and spatially coherent latent organization is important for the hierarchical SOL formulation: the first-order learner exposes an internally structured model state that can be used by the second-order controller to generate task-conditioned parameter transformations.

E.6 Task-Relevant Organization of Optimal-Path Nodes

The third visualization in Fig. 1 examines the latent positions of nodes belonging to the optimal path relative to the remaining maze nodes.

Shortest-path nodes occupy a compact and distinguishable region of the latent representation. In the reported qualitative clustering analysis, all shortest-path nodes fall within the same KMeans cluster. This organization demonstrates that the internal representation encodes task-relevant structure in addition to spatial organization.

The latent space therefore exhibits two complementary forms of structure:

1. **Environmental organization:** latent relationships reflect the spatial organization of the maze.
2. **Functional organization:** nodes associated with the shortest-path objective occupy a coherent region of the learned representation.

The coexistence of these properties is particularly relevant to second-order learning. The controller operates on the state of a first-order learner whose internal representation already reflects both the structure of the environment and the prediction objective. This creates an informative basis for producing parameter transformations when the graph is structurally perturbed.

E.7 Quantitative Spatial Coherence

The qualitative organization visible in Fig. 1 is supported by the quantitative pairwise-distance analysis conducted directly in the original latent space.

For the principal representation result reported in the main paper, the correlations between latent-space distance and maze-space distance are

$$r_p = 0.9042, \qquad r_s = 0.9085. \tag{27}$$

Both coefficients are close to one. The Pearson coefficient indicates strong agreement between the magnitudes of latent and environmental distances, while the Spearman coefficient indicates strong preservation of their relative ordering.

The close agreement between these two independently computed measures provides strong quantitative evidence that the first-order representation preserves the relational geometry of the environment. The result therefore goes beyond the visual appearance of the two-dimensional projection: the measured alignment is present in the original learned representation.

The correspondence between latent and environmental distances forms the quantitative basis of the environment–cognition isomorphism analysis in the paper. Together with the functional organization of path-relevant nodes, it shows that the first-order learner develops a representation that is simultaneously geometrically structured and aligned with the task objective.

Table 7: First-layer environment–representation distance correlations across the five independent large-scale Experiment 1 runs.

Seed	Pearson	Spearman
0	0.8590	0.8718
1	0.8554	0.8621
2	0.8761	0.8802
3	0.7274	0.7341
4	0.8142	0.8266
Mean	**0.8264**	**0.8350**

E.8 Large-Scale Stability Across Independent Seeds

The large-scale Experiment 1 implementation provides an additional representation diagnostic by measuring first-layer spatial alignment independently across the five experiment seeds. The resulting correlations are reported in Table 7.

Positive and substantial correlations are observed for every independent run. Across the five seeds, the mean Pearson correlation is 0.8264 and the mean Spearman correlation is 0.8350.

This repeated observation across independent model initializations and independently generated task collections strengthens the principal representation result. Environment-aligned latent geometry is therefore not restricted to the single representation used for the headline visualization; it is repeatedly observed within the large-scale experimental setting.

The agreement between Pearson and Spearman correlations across the five runs also indicates that both metric and ordinal aspects of spatial organization are consistently represented. This provides additional evidence that structured environmental organization is a recurring property of the learned first-order state.

E.9 Structured Representation as a Basis for Second-Order Adaptation

The representation results provide a direct connection between the two levels of the SOL hierarchy. At the first level, the GCN transforms graph-structured input into a latent state that preserves substantial environmental geometry while organizing nodes according to task-relevant path structure. At the second level, the controller operates on the state of this first-order system to produce a task-conditioned transformation of its parameters.

This organization provides a useful representational basis for rapid adaptation. When graph connectivity changes, the second-order learner does not need to reconstruct the learning process from the beginning or perform an iterative test-time optimization procedure. Instead, it acts on an already structured first-order model state and directly generates an adapted parameterization.

The representation analysis therefore complements the performance results of Experiment 1. The robustness experiments show that SOL maintains strong predictive performance under increasing structural perturbation, while the latent-space analysis shows that the underlying first-order learner possesses a highly organized internal representation of the environment. Together, these results support the proposed hierarchical interpretation of SOL: structured first-order representations provide the substrate on which a learned second-order adaptation mechanism can operate efficiently.

This relationship is examined from the opposite direction in Experiment 5, where the correspondence between meaningful node information and environmental geometry is deliberately disrupted. That experiment provides a complementary control for the link between representational structure and effective second-order adaptation.

F Extended Experiment 2: Meta-Learning Baselines and Computational Efficiency

This section expands the comparison between the proposed SOL Controller-GCN and established adaptation baselines under out-of-distribution structural perturbation. The main paper reports the predictive-performance comparison; here we provide additional details on the OOD protocol, adaptation procedures, and test-time computational requirements. These additional analyses highlight a key property of SOL: once the second-order controller has been learned, adaptation to an unseen task is produced directly without an iterative test-time optimization loop.

F.1 OOD Evaluation Protocol

All methods are evaluated under the same out-of-distribution perturbation split. Training tasks are generated using lower edge-removal probabilities,

$$p_{\text{train}} \in \{0.1, 0.2, 0.3\}, \tag{28}$$

whereas evaluation is performed on more strongly perturbed, previously unseen mazes,

$$p_{\text{test}} \in \{0.4, 0.5\}. \tag{29}$$

The evaluation therefore requires each method to transfer from the structural perturbation regime encountered during training to a harder perturbation regime at test time.

The experiment is repeated across five random seeds. For each seed, 50 training tasks are generated and performance is evaluated on 200 unseen test tasks. This produces a large-scale OOD comparison in which all methods are evaluated under the same task-generation and test conditions.

F.2 Adaptation Procedures

The comparison includes a non-adaptive baseline, ordinary fine-tuning, two gradient-based meta-learning methods, and the proposed second-order controller. The methods differ primarily in how they respond to a previously unseen test environment.

- **Vanilla GCN:** the pretrained GCN is evaluated directly on the unseen task without any parameter adaptation.
- **Fine-tuned GCN:** the pretrained GCN is adapted using five gradient-based optimization steps on the test task. This procedure requires test-task labels and backpropagation during adaptation.
- **MAML-GCN:** the GCN is meta-trained to support rapid gradient-based adaptation and is evaluated using five inner-loop gradient steps on each unseen test task.
- **Reptile-GCN:** the model is meta-trained using Reptile and is evaluated using five gradient-based inner-loop adaptation steps.
- **SOL Controller-GCN:** the learned second-order controller produces a task-conditioned GCN parameter transformation through a forward computation. The OOD meta-learning implementation uses an update scale of $\alpha = 0.1$. Application of the learned update requires neither test-time gradient computation nor labeled test examples.

The meta-learning implementation uses 100 meta-training epochs for MAML and 100 for Reptile. The base GCN is trained for 50 epochs, while the SOL controller is trained for 30 epochs. MAML uses one inner-loop step during meta-training and five adaptation steps during evaluation. Reptile uses five inner steps during both training and evaluation.

The distinction between SOL and the gradient-based baselines is therefore not whether learning uses gradients during the training phase. Rather, the distinguishing feature is the mechanism used after training when an unseen environment is encountered. MAML, Reptile, and ordinary fine-tuning perform a new optimization procedure for the test task, whereas SOL applies the previously learned controller directly to generate the adapted parameterization.

Table 8: Mean test-time adaptation cost and adaptation requirements in the OOD meta-learning comparison. SOL generates its task-conditioned update without test-time gradients or labeled test examples.

Method	Adaptation time (s)	Test gradients	Test labels
Vanilla GCN	0.0000	No	No
SOL Controller-GCN	0.0142	No	No
Fine-tuned GCN (5 steps)	0.0472	Yes	Yes
MAML-GCN (5 steps)	0.0498	Yes	Yes
Reptile-GCN (5 steps)	0.0495	Yes	Yes

F.3 OOD Generalization Results

The principal predictive-performance comparison is reported in Table 2 of the main paper and is not reproduced here. Under the held-out perturbation regime $p_{\text{test}} \in \{0.4, 0.5\}$, the SOL Controller-GCN achieves $79.33\% \pm 0.67\%$ accuracy. MAML-GCN obtains $78.66\% \pm 0.95\%$, Reptile-GCN obtains $76.38\% \pm 4.46\%$, the fine-tuned GCN obtains $62.30\% \pm 6.92\%$, and the Vanilla GCN obtains $57.78\% \pm 8.38\%$.

The corresponding BCE losses are 0.511 for SOL, 0.504 for MAML-GCN, 0.582 for Reptile-GCN, 0.830 for fine-tuning, and 0.965 for the Vanilla GCN. Thus, SOL reaches the same high-performance regime as the gradient-based meta-learning approaches while using a fundamentally different test-time adaptation mechanism.

The paired comparison between SOL and the Vanilla GCN yields $p = 0.005721$ for accuracy and $p = 0.018265$ for BCE loss in the existing five-seed evaluation. These results establish that the improvement observed under the OOD perturbation split is reproducible across the repeated large-scale evaluation.

The predictive results are especially relevant when considered together with the adaptation procedure. SOL achieves its OOD performance without performing an inner-loop optimization on the test task and without requiring labeled test examples. The learned second-order controller therefore converts experience across training environments into a reusable task-conditioned adaptation rule that can be applied directly to an unseen structural configuration.

F.4 Adaptation-Time Comparison

In addition to predictive performance, the implementation records the test-time adaptation cost of each method. These measurements are summarized in Table 8.

The timing results expose the computational consequence of the different adaptation mechanisms. SOL requires approximately 0.0142 seconds per adaptation in this implementation, compared with approximately 0.047–0.050 seconds for the five-step gradient-based procedures. This corresponds to an approximately $3.5\times$ reduction in adaptation time relative to the gradient-based methods measured in the same experiment.

The source of this difference is structural. Fine-tuning, MAML, and Reptile must repeatedly evaluate gradients and update task-specific parameters during test-time adaptation. SOL instead evaluates the trained controller and constructs the adapted GCN parameters directly. Consequently, its test-time adaptation consists of a learned forward computation rather than an iterative optimization process.

Figure 5 visualizes the joint relationship between OOD classification accuracy and mean test-time adaptation time across the evaluated methods. SOL attains $79.33\% \pm 0.67\%$ accuracy at an average adaptation time of 0.0142 s, while the gradient-based adaptation methods require approximately 0.047–0.050 s. The vertical error bars show the reported accuracy standard deviation across five seeds.

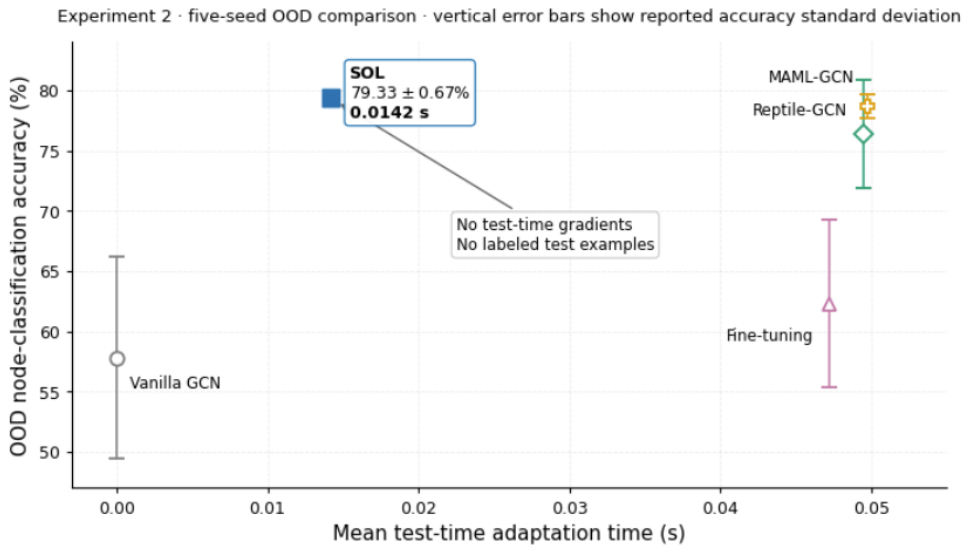


Figure 5: OOD node-classification accuracy versus mean test-time adaptation time for the Experiment 2 comparison. Vertical error bars show the reported standard deviation in accuracy across five seeds. SOL combines $79.33\% \pm 0.67\%$ OOD accuracy with a mean adaptation time of 0.0142 s and requires neither test-time gradients nor labeled test examples. Fine-tuning, MAML-GCN, and Reptile-GCN use five gradient-based adaptation steps.

F.5 Performance and Adaptation Efficiency

The OOD comparison therefore evaluates two complementary properties of the adaptation mechanism: predictive transfer to harder unseen perturbations and the computation required to obtain that adaptation. The main-paper results show that SOL remains in the same high-accuracy regime as MAML under the held-out structural shift, while Table 8 shows that the SOL update is produced substantially faster in the implemented comparison.

This combination is a central feature of the proposed second-order learning framework. During training, the controller learns how the state of the first-order model should be transformed across a distribution of perturbed tasks. At test time, that learned adaptation knowledge is reused directly. The cost of discovering an adaptation strategy is therefore shifted to the second-order training phase, allowing a novel task to be handled through a single learned transformation rather than through repeated gradient updates.

Experiment 2 thus demonstrates that SOL is not only capable of transferring to a harder OOD perturbation regime, but can do so using a test-time mechanism that requires no gradients and no labeled test data. The result complements Experiment 1 by showing that the learned second-order adaptation principle remains effective when the evaluation distribution is explicitly separated from the perturbation levels used for training.

G Extended Experiment 3: Enriched Latent Input for Controller Adaptation

Experiment 3 evaluates whether exposing the second-order learner to the first-order latent representation in addition to logits and current parameters provides a richer signal for adaptation.

G.1 Controller Input Construction

For N nodes, hidden dimension d_h, output dimension d_o, and GCN parameter count $|\theta|$, the enriched controller input dimension is

$$d_{\text{in}} = Nd_h + Nd_o + |\theta|. \tag{30}$$

The hidden activation and output are flattened and concatenated before the current parameter vector is appended.

The controller then applies

$$h_c = \text{LeakyReLU}(W_1 s + b_1), \tag{31}$$

$$\tilde{h}_c = \text{Dropout}(h_c), \tag{32}$$

$$\Delta\theta = W_2 \tilde{h}_c + b_2. \tag{33}$$

G.2 Reported Result

The enriched controller achieves 93.6% accuracy with BCE loss 0.2335 on the unseen-maze evaluation reported in the main paper. The accuracy is comparable to the corresponding small-scale adaptation result, while the lower loss indicates more confident task-aligned predictions in this evaluation.

G.3 Controller Input Scaling

Equation equation 30 also makes the principal scaling property of the flattened implementation explicit. The latent and output portions grow linearly with the number of nodes N, and the first controller linear layer therefore grows with N as well. This accounting is useful when interpreting the maze-size study: the GCN representation can be evaluated on larger graphs, but the flattened controller itself is tied to the input size for which its first linear layer was instantiated.

This appendix reports this property as an implementation characteristic of the current method; the paper's size-generalization experiment concerns the transfer of spatial representation.

H Extended Experiment 4: Generalization Across Maze Sizes

Experiment 4 examines whether the spatial organization learned by the first-order GCN on smaller maze environments is retained when the same model is evaluated on larger grids. The analysis uses the Pearson correlation between pairwise Euclidean distances in the first-layer latent representation and the corresponding distances in the maze as the measure of spatial alignment.

The headline cross-size results are reported in Experiment 4 of the main paper. Rather than reproducing those values in a second table, this section provides additional detail on the evaluation protocol and the relationship between the two training conditions.

H.1 Transfer from 8×8 to Larger Grids

In the first evaluation condition, the GCN is trained on 8×8 maze environments and then evaluated directly on larger, unblocked grids without additional size-specific training. Evaluation is performed on 10×10, 12×12, and 14×14 mazes.

As reported in the main paper, the corresponding Pearson correlations are 0.9462, 0.9559, and 0.9623, respectively. The correlations therefore remain high as the evaluation grid increases substantially beyond the training size.

This result shows that the organization of the first-layer representation is not restricted to the spatial extent of the training grid. The same first-order model retains strong correspondence between latent-space distances and environmental distances when applied to larger graph instances.

An additional feature of the result is that spatial alignment does not deteriorate as maze size increases over the tested range. The correlation changes from 0.9462 at 10×10 to 0.9623 at 14×14.

Thus, the larger evaluation graphs do not disrupt the spatial organization measured in the learned representation.

H.2 Effect of Increased Training-Size Diversity

The second condition introduces additional size variation during training by using both 8×8 and 10×10 mazes. The resulting model is then evaluated on the larger 12×12 and 14×14 environments.

The main-paper results report Pearson correlations of 0.9570 and 0.9631 at these two evaluation sizes. For comparison, training only on 8×8 mazes yields correlations of 0.9559 and 0.9623 at the same respective sizes.

The differences between the two training conditions are therefore small: adding 10×10 training examples increases the measured correlation by 0.0011 at 12×12 and by 0.0008 at 14×14. Both training conditions consequently produce strongly spatially aligned representations on grids larger than those used in the original 8×8 training setting.

These results indicate that the cross-size spatial organization is already present when training is restricted to the smaller grid. Introducing additional training-size diversity preserves this behavior and produces a small further increase in the measured alignment.

H.3 Relation to the SOL Hierarchy

Experiment 4 characterizes the transfer properties of the first-order representation. The result complements the adaptation experiments by showing that the GCN's learned spatial organization extends beyond a single graph size: pairwise relationships in the latent representation remain strongly aligned with environmental geometry on larger maze instances.

This distinction is useful within the SOL hierarchy. The first-order GCN provides the structured representation of the environment, while the second-order controller operates on the state of that first-order learner to produce task-conditioned parameter transformations. Experiment 4 demonstrates that the spatial organization of the first component remains transferable across the tested maze sizes.

The experiment therefore extends the representation analysis beyond changes in graph connectivity. Whereas Experiment 1 evaluates adaptation under structural perturbation at a fixed maze scale, Experiment 4 shows that the underlying first-order spatial representation remains strongly organized when the graph itself is evaluated at larger spatial scales.

I Extended Experiment 5: Representation Collapse in Non-Isomorphic Mazes

Experiment 5 provides a targeted control for the representational-isomorphism analysis. The experiment tests whether the strong correspondence between the first-order latent representation and the external maze depends on meaningful feature–geometry structure.

The graph connectivity and shortest-path prediction task are preserved, while the spatially meaningful node features are replaced with i.i.d. Gaussian noise. The underlying graph problem therefore remains intact, but the systematic correspondence between node features and environmental geometry is disrupted.

This control complements the positive representation results reported in Sections E and H: rather than asking whether structured representations transfer across perturbations or graph sizes, Experiment 5 examines what happens when the feature–geometry relationship supporting that structure is deliberately removed.

I.1 Construction of the Non-Isomorphic Condition

For each generated maze, the standard spatially meaningful feature matrix is replaced with random features sampled independently for each node. The graph connectivity and shortest-path task are otherwise unchanged.

Table 9: Spearman distance correlations for the three non-isomorphic mazes. Values close to zero indicate that the ordering of spatial relationships is no longer preserved in the learned latent representation.

Maze	Spearman correlation
1	-0.0350
2	-0.0599
3	-0.0615
Mean	-0.0521

The manipulation therefore preserves the combinatorial structure of the maze while removing the systematic spatial information carried by the standard node features. This provides a controlled way to examine the relationship between meaningful environmental structure, the geometry of the learned first-order representation, and the effectiveness of second-order adaptation.

I.2 Main-Paper Non-Isomorphic Results

The principal non-isomorphic results are reported in Table 3 and Fig. 2 of the main paper and are not reproduced here.

Across the three reported non-isomorphic mazes, the Pearson correlations between latent-space distance and environmental distance are close to zero. The main-paper analysis summarizes the contrast with the isomorphic condition as a reduction in spatial alignment from approximately 0.86 to approximately 0.05, together with a reduction in task performance relative to the isomorphic setting.

The corresponding t-SNE visualizations in Fig. 2 provide a qualitative view of the same effect. The organized spatial structure visible in the isomorphic representation is strongly disrupted after the meaningful feature–geometry correspondence is removed.

Because the graph connectivity and prediction objective are retained in this control, the change in representational organization is associated specifically with the disruption of the structured relationship between the node features and the external environment.

I.3 Additional Rank-Based Alignment Diagnostic

In addition to the Pearson correlations reported in the main-paper table, the existing analysis records Spearman correlations between pairwise latent-space distances and maze-space distances. These values provide an additional rank-based measure of representational alignment.

The Spearman coefficients are close to zero for all three non-isomorphic mazes, with a mean of approximately -0.0521. This agrees closely with the near-zero Pearson correlations reported in Table 3 of the main paper.

The agreement between the two correlation measures is informative because they capture different properties of the latent geometry. Pearson correlation measures correspondence in distance magnitude, whereas Spearman correlation measures preservation of distance ordering. Their simultaneous reduction toward zero shows that disrupting feature–geometry correspondence removes both the approximately metric and the ordinal spatial organization observed in the isomorphic representation.

I.4 Relation to the Positive Isomorphism Result

Experiment 5 provides a complementary counterpart to the positive representation results. In the standard isomorphic setting, the first-order GCN develops a strongly spatially organized latent representation, with the principal analysis reporting Pearson and Spearman correlations of 0.9042 and 0.9085, respectively. The large-scale analysis further shows that positive environment–representation alignment is repeatedly observed across independent seeds.

Under the non-isomorphic manipulation, both Pearson and Spearman spatial alignment instead move close to zero. The contrast therefore extends beyond the appearance of the t-SNE projections: it is also present in quantitative distance-based measurements computed from the learned representation.

This controlled contrast strengthens the connection between meaningful feature–geometry correspondence and the emergence of an environmentally organized first-order latent space.

I.5 Interpretation of the Control

The non-isomorphic experiment isolates an important component of the SOL hierarchy. In the standard setting, the first-order learner develops an internal representation whose geometry strongly corresponds to the structure of the external environment. The second-order controller can then operate on the state of this structured first-order system when generating task-conditioned parameter transformations.

When meaningful feature–geometry correspondence is deliberately disrupted, the spatial organization of the first-order representation is correspondingly reduced. The associated decrease in performance relative to the isomorphic condition links representational structure to the effectiveness of the adaptation process.

Experiment 5 therefore provides a controlled counterpart to the positive results of Experiments 1 and 4. The positive experiments show the emergence, robustness, and transfer of structured first-order representations, while the non-isomorphic control shows that removing meaningful environmental correspondence substantially disrupts that structure. Together, these results support the central SOL interpretation that effective second-order adaptation is associated with a first-order representation that is meaningfully organized with respect to the environment.

J Extended Experiment 6: Real-World OSM Road-Network Adaptation

Experiment 6 evaluates the SOL adaptation mechanism on a real-world road network extracted from OpenStreetMap. In this setting, road closures induce structural distribution shift while the underlying shortest-path prediction objective remains analogous to the synthetic-maze experiments.

The principal predictive-performance comparison is reported in Table 4 of the main paper. This appendix therefore focuses on additional experimental details that are not reproduced there, including the road-network construction procedure, closure-task generation, training configuration, complementary path-recovery metrics, and test-time adaptation cost.

J.1 Road-Network Construction

The executed experiment uses a walking network extracted from OpenStreetMap for Piedmont, California. From the downloaded road graph, the implementation constructs an undirected connected subgraph containing 250 nodes and 322 edges. Nodes represent road-network intersections and edges represent traversable road segments.

When the downloaded graph contains more nodes than the requested experimental budget, the implementation selects a connected local region around a sampled center node. The resulting subgraph is then relabeled for tensor construction.

Shortest-path targets are computed using road-segment length. The task therefore retains the path-planning objective used in the synthetic maze experiments while replacing the regular grid with an irregular graph derived from a real transportation network.

J.2 Road-Closure Tasks

Structural perturbations are introduced by removing road-network edges while retaining usable source-to-goal shortest-path tasks. The executed experiment uses closure rates

$$c \in \{0.05, 0.10, 0.15, 0.20\}. \tag{34}$$

Table 10: Additional positive-class path-recovery metrics from the existing OSM evaluation (mean $\pm$ standard deviation over the 80 unseen road-closure tasks). Accuracy, BCE loss, and F1 are reported in Table 4 of the main paper.

Model	Precision	Recall	Path IoU
Vanilla GCN	0.0823 ± 0.0268	1.0000 ± 0.0000	0.0823 ± 0.0268
SOL Controller-GCN	0.1260 ± 0.0753	0.4055 ± 0.2431	0.1077 ± 0.0667
MAML-GCN	$\mathbf{0.1375 \pm 0.0415}$	0.7696 ± 0.1406	$\mathbf{0.1324 \pm 0.0401}$
Reptile-GCN	0.1140 ± 0.0620	0.7412 ± 0.1970	0.1101 ± 0.0589

The SOL controller is trained on 40 road-closure tasks and evaluated on 80 previously unseen closure tasks.

This evaluation extends the structural-perturbation setting of Experiment 1 from regular synthetic grids to an irregular real-world graph. The resulting tasks vary in both the location and severity of road closures, requiring the adaptation mechanism to respond to changes in the connectivity of a realistic network.

J.3 Training Configuration

The executed OSM configuration uses:

- 200 epochs for the base GCN;
- 50 epochs for the SOL controller;
- 50 meta-training epochs for MAML-GCN;
- 50 meta-training epochs for Reptile-GCN;
- an SOL parameter-update scale of $\alpha = 0.1$;
- one MAML inner-loop step during meta-training and five gradient-based adaptation steps during evaluation; and
- five Reptile inner-loop steps during both training and evaluation.

The supplied implementation uses Adam-based optimization for the base and meta-learning models. The SOL controller uses learning rate 0.001 and weight decay 10^{-4} in the OSM experiment.

Once controller training is complete, the learned SOL mapping is applied directly to each unseen road-closure task. As in the synthetic experiments, this test-time application does not require gradient computation or labeled examples from the new task.

J.4 Predictive Performance and Additional Path-Recovery Metrics

The primary comparison in Table 4 of the main paper shows that SOL achieves the highest reported overall node-classification accuracy, $75.72\% \pm 4.93\%$, compared with $63.25\% \pm 5.58\%$ for MAML-GCN, $53.14\% \pm 8.20\%$ for Reptile-GCN, and $16.53\% \pm 9.43\%$ for the Vanilla GCN.

The executed evaluation additionally records precision, recall, and path IoU. These complementary path-recovery metrics are reported below without reproducing the Accuracy, BCE, and F1 columns already presented in the main paper.

The additional metrics provide a complementary view of performance on the road-network task. SOL achieves the strongest overall node-wise accuracy in the reported comparison, while the methods exhibit different trade-offs in their recovery of the positive shortest-path class. MAML obtains the highest precision and path IoU and also achieves higher recall, whereas SOL's principal advantage in this experiment is its substantially higher overall classification accuracy together with direct gradient-free adaptation.

These results show that the real-world comparison contains two distinct evaluation perspectives. Accuracy measures prediction over the complete road graph, whereas precision, recall, F1, and IoU characterize recovery of the smaller set of nodes belonging to the shortest path. Reporting both

Table 11: Test-time adaptation cost in the executed OSM road-network experiment.

Method	Adaptation time (s)	Test gradients	Test labels
Vanilla GCN	0.0000	No	No
SOL Controller-GCN	0.0077	No	No
MAML-GCN (5 steps)	0.0185	Yes	Yes
Reptile-GCN (5 steps)	0.0195	Yes	Yes

views provides a more complete description of behavior under road closures while preserving the main result that SOL achieves the highest overall accuracy in the experiment.

J.5 Computational Efficiency on the Road Graph

The executed experiment also records the wall-clock time required for test-time adaptation. Table 11 reports these measurements together with the test-time requirements of the evaluated methods.

SOL performs adaptation in approximately 0.0077 seconds in this executed comparison. The five-step MAML and Reptile procedures require approximately 0.0185 and 0.0195 seconds, respectively. SOL therefore performs its learned adaptation approximately 2.4–$2.5\times$ faster than the gradient-based meta-learning procedures measured in the same experiment.

The difference follows directly from the adaptation mechanism. MAML and Reptile perform iterative gradient-based updates for each new road-closure task, whereas SOL evaluates the trained second-order controller and generates the task-conditioned first-order parameter transformation directly.

The real-world experiment therefore preserves a central computational property of SOL: adaptation to an unseen environment is performed without an iterative test-time optimization loop, test-time gradients, or labeled examples from the new task.

Figure 6 combines this computational comparison with the complementary predictive metrics reported in Section J.4. Panel (a) shows the precision–recall operating points for recovery of the positive shortest-path class. The evaluated methods occupy different regions of the precision–recall space: MAML and Reptile obtain higher path recall than SOL, whereas SOL attains the highest overall node-wise classification accuracy in the corresponding aggregate comparison.

Panel (b) places overall accuracy alongside F1 and path IoU from the same executed evaluation. This makes explicit the distinction between performance over all road-network nodes and recovery of the smaller positive shortest-path class. The error bars in panels (a) and (b) denote the standard deviation across the 80 unseen road-closure tasks and should therefore be interpreted as task-level variability rather than uncertainty across independently trained models.

Panel (c) shows the corresponding mean adaptation time per task. SOL requires approximately 7.7 ms, compared with 18.5 ms for MAML and 19.5 ms for Reptile. Vanilla GCN incurs no adaptation cost because its parameters are left unchanged. The predictive and timing views therefore characterize two complementary properties of the real-world experiment: behavior on the shortest-path prediction task and the computation required to obtain an adapted model.

J.6 Performance as a Function of Road-Closure Rate

The existing OSM evaluation also records performance separately at each road closure rate, $c \in \{0.05, 0.10, 0.15, 0.20\}$. Accuracy, BCE loss, precision, recall, F1, and path IoU are tracked as the road network becomes progressively more disrupted.

This decomposition complements the aggregate results in Table 4 by exposing adaptation behavior as a function of structural-perturbation severity. It provides the real-road-network analogue of the perturbation-strength analysis used in Experiment 1 and allows the stability of the different adaptation procedures to be examined across increasing levels of road closure.

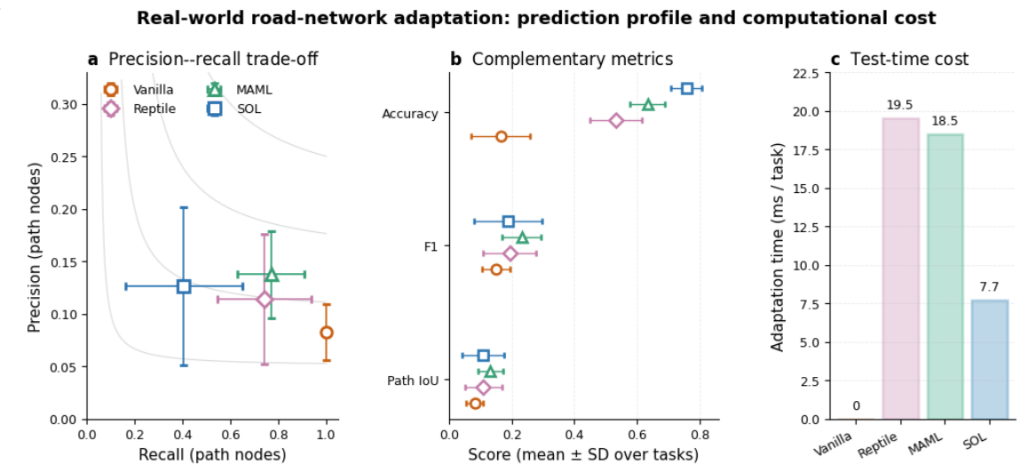


Figure 6: **Prediction characteristics and computational cost on the real-world OSM road-network task.** (a) Precision–recall operating points for shortest-path recovery across the 80 unseen road-closure tasks. Horizontal and vertical error bars denote the standard deviation across tasks. (b) Overall accuracy, F1 score, and path IoU from the same evaluation, shown as mean ± standard deviation across tasks. (c) Mean test-time adaptation cost per task. SOL applies the learned second-order controller directly, whereas MAML and Reptile use five gradient-based adaptation steps. Vanilla GCN performs no test-time adaptation. All quantities are taken from the same executed OSM evaluation reported in Table 4 and the corresponding appendix analysis.

J.7 OSM Training Dynamics

The existing implementation also records optimization trajectories for the base GCN, SOL controller, MAML-GCN, and Reptile-GCN. These curves characterize the training processes that precede evaluation on the unseen road-closure tasks.

For SOL, the controller-training trajectory is particularly useful for illustrating the separation between second-order learning and subsequent adaptation. Gradients are used during the training phase to learn the controller parameters. Once training is complete, the learned controller is applied directly to new road-closure tasks without an additional gradient-based optimization procedure.

J.8 Real-World Extension of the SOL Adaptation Mechanism

Experiment 6 extends the same hierarchical adaptation mechanism from regular synthetic maze graphs to an irregular real-world road network. The first-order GCN operates on the road graph, while the second-order controller learns a task-conditioned transformation of the first-order model under structural changes induced by road closures.

The results show that this adaptation principle remains effective after the transition to the real-world graph domain. SOL achieves the highest overall node-classification accuracy in the reported comparison while retaining single-pass, gradient-free, and label-free adaptation at test time. The measured adaptation times further show that the learned controller can generate the adapted model substantially faster than the five-step gradient-based MAML and Reptile procedures used in the same evaluation.

Experiment 6 therefore provides an important extension of the synthetic results: the proposed second-order adaptation mechanism is not restricted to regular grid graphs, but also operates effectively on structurally perturbed road-network graphs derived from real geographic data.

K Extended Experiment 7: Second-Order Learning for Value Adaptation

Experiment 7 evaluates whether the SOL hierarchy extends beyond binary shortest-path classification to a node-wise regression problem. In this setting, the first-order GCN learns a cumulative value function over the grid, while the second-order controller adapts the first-order model when the reward landscape changes.

The main paper reports the headline value-adaptation results. This appendix provides the corresponding reward construction, dynamic-programming targets, perturbation procedure, controller-training protocol, metric definitions, and a consolidated quantitative comparison.

K.1 Reward Construction

The value environment is defined on an $n \times n$ grid. Each location (i, j) is assigned the deterministic base reward

$$R(i, j) = i + j. \tag{35}$$

The node feature vector is

$$x_{i,j} = [i, j, R(i, j)]. \tag{36}$$

The executed value-adaptation experiment uses a 5×5 grid and a first-order GCN with hidden dimension 16. In contrast to the binary path-membership experiments, the GCN output in this setting is interpreted as a continuous estimate of the cumulative value associated with each node.

K.2 Dynamic-Programming Value Targets

Ground-truth value targets are generated using dynamic programming. The terminal state is the lower-right grid location, with

$$V(n-1, n-1) = R(n-1, n-1). \tag{37}$$

For the remaining states, the optimal value is computed recursively as

$$V(i, j) = R(i, j) + \max\left(V(i+1, j), V(i, j+1)\right), \tag{38}$$

over valid downward and rightward transitions.

The first-order value GCN is trained to approximate these dynamic-programming targets using mean-squared error,

$$\mathcal{L}_{\text{value}} = \frac{1}{N} \sum_{i=1}^{N} \left(V_i - \hat{V}_i\right)^2. \tag{39}$$

This formulation gives the first-order learner a regression objective whose predictions can also be converted into decisions by selecting the successor state with the larger predicted value.

K.3 Reward Perturbations

To create changing reward environments, the base rewards are modified using a multiplicative sign mask,

$$R'(i, j) = R(i, j)m(i, j), \qquad m(i, j) \in \{+1, -1\}. \tag{40}$$

Table 12: Value-adaptation performance under reward perturbation. The adapted SOL model substantially improves value prediction and the policy induced by the predicted value function.

Model	MSE	MAE	R^2	Policy accuracy
Unadapted GCN	297.47	14.73	-2.16	25.0%
Adapted GCN (SOL)	**33.50**	**4.99**	**0.64**	**66.7%**

The probability of assigning the negative sign is selected from

$$p \in \{0.1, 0.2, 0.3\}. \tag{41}$$

These perturbations alter the value landscape while preserving the underlying grid topology. Consequently, the task tests whether the second-order learner can reconfigure the first-order value estimator in response to changes in the reward structure rather than changes in graph connectivity.

This experiment therefore complements the earlier structural-perturbation studies. Experiments 1 and 6 modify graph connectivity, whereas Experiment 7 changes the task's reward landscape while retaining the same underlying graph.

K.4 Value Controller Training

The value controller receives information from the current first-order model, including its value predictions and current GCN parameters, and produces the task-conditioned parameterization used for adaptation to a perturbed reward environment.

The base value GCN is first trained for 50 epochs with learning rate 0.01. Five perturbed reward tasks are then generated for second-order controller training. In the final metric-producing implementation, the value controller is trained for 100 epochs with learning rate 0.01.

The purpose of the second-order training phase is to learn how changes in the state of the first-order value model should be translated into an adapted parameterization. Once learned, this mapping can be applied to a changed reward environment without retraining the first-order value GCN from the beginning.

K.5 Evaluation Metrics

The value-adaptation experiment evaluates both regression quality and the decision behavior induced by the predicted value function.

- **Mean-squared error (MSE):** measures the squared deviation between predicted and target node values.
- **Mean-absolute error (MAE):** measures the average absolute deviation between predicted and target values.
- **Coefficient of determination (R^2):** measures the proportion of variation in the target value function captured by the predictions.
- **Policy accuracy:** measures the fraction of decision states at which the action implied by the predicted value function agrees with the action implied by the dynamic-programming target values.

The first three metrics quantify the fidelity of the learned value function, while policy accuracy measures whether that value representation induces the same local decision as the dynamic-programming solution.

K.6 Value-Adaptation Results

The headline numerical results are summarized in Table 12. The same values are reported in prose in the main paper; the table provides a consolidated view of all four evaluation metrics.

Table 13: Performance on 20 newly generated held-out reward-perturbation tasks using the already-trained Experiment 7 models. Values are mean ± standard deviation across the 20 evaluation tasks.

Metric	Unadapted GCN	SOL-adapted GCN
MSE	315.47 ± 34.14	$\mathbf{91.31 \pm 32.61}$
MAE	15.03 ± 1.32	$\mathbf{7.79 \pm 1.52}$
R^2	-3.56 ± 1.76	$\mathbf{-0.29 \pm 0.76}$
Policy accuracy	$10.21\% \pm 9.70\%$	$\mathbf{70.21\% \pm 7.68\%}$

Adaptation reduces MSE from 297.47 to 33.50, corresponding to an approximately 88.7% reduction in squared prediction error. MAE decreases from 14.73 to 4.99, an approximately 66.1% reduction in absolute prediction error.

The coefficient of determination increases from -2.16 before adaptation to 0.64 after adaptation. Thus, the adapted value model moves from poor agreement with the dynamic-programming value targets to a representation that captures a substantial proportion of their variation.

The improvement also transfers to the decisions induced by the learned value function. Policy accuracy increases from 25.0% to 66.7%, an absolute gain of 41.7 percentage points. The second-order transformation therefore improves not only numerical value prediction but also the action choices derived from those predictions.

These results extend the SOL mechanism beyond the binary node-classification setting. The second-order learner can adapt a first-order regression model when the quantity being represented—here, the cumulative reward landscape—changes substantially.

K.7 Held-Out Reward-Task Generalization

The headline value-adaptation result reported in the main paper is complemented here with an additional held-out evaluation. After training the base value GCN and second-order controller using the existing Experiment 7 procedure, the trained models are held fixed and evaluated on 20 newly generated reward perturbations that are not used during controller training.

The held-out tasks use the same 5×5 grid, reward construction, and perturbation family as the original experiment, with sign-perturbation probability

$$p \in \{0.1, 0.2, 0.3\}.$$

No additional optimization of either the first-order GCN or second-order controller is performed on these evaluation tasks.

Table 13 summarizes the resulting regression and decision-level performance.

SOL improves all four metrics on every one of the 20 held-out tasks. Relative to the unadapted GCN, the mean task-level reduction is approximately 70.5% for MSE and 47.5% for MAE. Policy accuracy increases by an average of 60.0 percentage points.

Figure 7 shows the paired task-level results. Each gray line connects the unadapted and SOL-adapted performance for the same held-out reward perturbation. The consistent direction of the paired changes shows that the aggregate improvement is not driven by a small number of individual tasks.

Paired nonparametric tests provide the same conclusion. Two-sided Wilcoxon signed-rank tests give $p = 1.91 \times 10^{-6}$ for MSE, MAE, and R^2, and $p = 8.68 \times 10^{-5}$ for policy accuracy. SOL improves MSE, MAE, R^2, and policy accuracy on all $20/20$ evaluated tasks.

The held-out R^2 result also provides a useful distinction between relative adaptation benefit and absolute regression quality. Although mean R^2 remains slightly below zero after adaptation, it improves substantially from -3.56 to -0.29, while error metrics decrease markedly and policy agreement increases to approximately 70%. Thus, across newly generated reward perturbations, the learned second-order transformation consistently produces a more useful value representation than the unadapted first-order model.

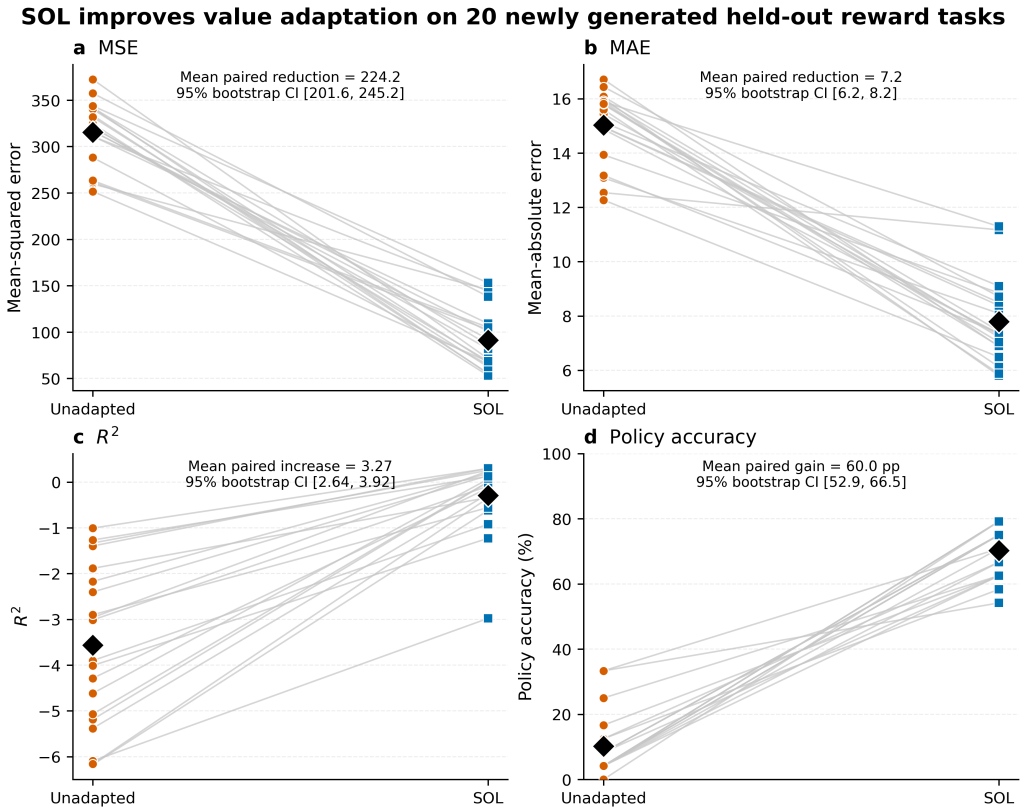


Figure 7: **Held-out value adaptation across 20 newly generated reward-perturbation tasks.** Each gray line connects the unadapted and SOL-adapted result for the same evaluation task, and black diamonds denote task-level means. (a) MSE decreases by a mean paired amount of 224.2, with a 95% bootstrap confidence interval of $[201.6, 245.2]$. (b) MAE decreases by 7.2 on average, with a 95% bootstrap confidence interval of $[6.2, 8.2]$. (c) R^2 increases by a mean of 3.27, with a 95% bootstrap confidence interval of $[2.64, 3.92]$. (d) Policy accuracy increases by 60.0 percentage points, with a 95% bootstrap confidence interval of $[52.9, 66.5]$. Confidence intervals are computed over paired task-level differences from this fixed trained controller and therefore quantify variability across held-out tasks rather than across independently trained model seeds.

This additional evaluation strengthens the generalization interpretation of Experiment 7 without introducing a new task family or adaptation procedure. The same trained second-order controller is applied directly to reward environments generated from the original perturbation distribution, and the resulting improvements extend consistently across previously unused tasks.

K.8 Value-Task Training Dynamics and Visual Diagnostics

The existing implementation additionally records several diagnostics from the value-adaptation experiment. These include the base-GCN MSE training trajectory, controller adaptation-loss trajectory, comparisons of adapted and unadapted MSE, R^2, and policy accuracy, and visualizations of the predicted value function.

These outputs provide complementary views of the value experiment. The training curves characterize optimization of the first- and second-order learners, while the value-prediction visualizations show how the learned function changes after adaptation to the perturbed reward landscape.

K.9 Extension of SOL Beyond Path Classification

Experiment 7 demonstrates that the SOL hierarchy can operate on a learning objective that differs substantially from shortest-path membership classification. The first-order learner now represents a continuous value function, the environmental change occurs through the reward landscape, and the quality of adaptation is measured using regression and decision-level metrics.

Despite this change in objective, the same hierarchical principle remains: a first-order model represents the current task, and a second-order learner uses the state of that model to generate a task-conditioned reparameterization when the environment changes.

The large reductions in MSE and MAE, the increase in R^2, and the 41.7-percentage-point improvement in policy accuracy show that the learned second-order transformation produces a substantially improved value representation under reward perturbation. Experiment 7 therefore broadens the empirical scope of SOL from structural graph adaptation to adaptation of a continuous value function under a changing reward landscape.

L Additional Metric Definitions for Expanded Diagnostics

The primary evaluation metrics used throughout the paper—including binary cross-entropy, node-classification accuracy, representational alignment, MSE, MAE, R^2, and policy accuracy—are defined in the original experimental-details appendix. To avoid repeating those definitions, this section records the additional path-recovery and computational diagnostics used in the expanded analyses.

L.1 Positive-Class Path-Recovery Metrics

For the shortest-path classification tasks, nodes belonging to the target path form the positive class. Let TP, FP, and FN denote the numbers of true-positive, false-positive, and false-negative node predictions, respectively.

Precision measures the fraction of predicted path nodes that belong to the ground-truth path:

$$\mathrm{Precision} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}}. \tag{42}$$

Recall measures the fraction of ground-truth path nodes recovered by the model:

$$\mathrm{Recall} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}}. \tag{43}$$

The F1 score is the harmonic mean of precision and recall:

$$\mathrm{F1} = \frac{2\,\mathrm{Precision}\,\mathrm{Recall}}{\mathrm{Precision} + \mathrm{Recall}}. \tag{44}$$

Path intersection-over-union (IoU) measures the overlap between the predicted positive node set and the ground-truth path:

$$\mathrm{IoU}_{\mathrm{path}} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP} + \mathrm{FN}}. \tag{45}$$

These metrics complement overall node-wise accuracy in the OSM experiment. Accuracy evaluates predictions over the complete road graph, whereas precision, recall, F1, and path IoU focus specifically on recovery of the shortest-path class. Reporting both provides a more complete characterization of adaptation behavior on the real-world graph.

Table 14: Mapping between the reported experiments and their corresponding implementation notebooks.

Experimental component	Implementation notebook
Experiments 1, 3, 4, and 5, together with the qualitative representation analyses	`SOL_baseExperiments_Maze.ipynb`
Large-scale Experiment 1 robustness and statistical validation	`SOL_LargeScale.ipynb`
Experiment 1 robustness figure and verification	`SOL_Experiment1_Robustness_Figure_Verification.ipynb`
Experiment 2 OOD comparison with fine-tuning, MAML, and Reptile	`SOL_meta_learning_comparisons_maze_env.ipynb`
Experiment 6 real-world OSM road-network adaptation	`SOL_MetalearningComparison_realGraph.ipynb`
Experiment 7 value-adaptation experiment	`SOL_baseExperiments_Maze.ipynb`
Experiment 7 held-out reward-task evaluation and paired analysis	`SOL_baseExperiments_Maze_test.ipynb`

L.2 Adaptation-Time Diagnostic

The computational-efficiency analyses report wall-clock adaptation time for the executed implementations. This quantity measures the time required to produce the task-adapted model after an unseen task is presented.

For SOL, this corresponds to applying the trained second-order controller and constructing the task-conditioned first-order parameterization. For gradient-based baselines, it includes the corresponding test-time inner-loop adaptation steps.

The timing measurements are therefore used as an implementation-level comparison of the computational cost of the evaluated adaptation procedures. All timing comparisons in the appendix are reported together with the number of test-time gradient steps and whether labeled test examples are required.

M Result Provenance and Notebook Mapping

For reproducibility, Table 14 maps each reported experimental component to the implementation notebook from which its results and diagnostics are obtained.

The mapping distinguishes the qualitative and small-scale maze analyses from the separate large-scale Experiment 1 validation, while preserving the experiment organization used throughout the paper. Several experiments share the same base notebook because they build on the same maze environment and first-order GCN implementation, whereas the large-scale statistical, meta-learning, and real-road-network evaluations are implemented in dedicated notebooks.

N Consolidated Hyperparameter and Protocol Summary

The preceding experiment sections provide the complete protocol for each evaluation. Table 15 consolidates the principal training configurations and task counts in a single location for reproducibility.

The consolidated view highlights that the same hierarchical organization is maintained across the experimental settings: a first-order model is trained for the underlying task, and a second-order controller or meta-learning procedure is subsequently trained across task variations. The experiments then differ in the source of distributional change—graph perturbation, held-out perturbation severity, real-world road closures, or changes to the reward landscape.

Table 15: Consolidated experimental configurations from the supplied implementations. The table summarizes the settings used for the principal large-scale maze, OOD meta-learning, real-world OSM, and value-adaptation experiments.

Experiment	Base learner	Controller / meta training	Task counts	Key evaluation settings
Large-scale maze	2-layer GCN, $5 \to 16 \to 1$; 50 epochs; Adam; lr $= 0.01$	SOL controller; 30 epochs; Adam; lr $= 0.01$; weight decay $= 10^{-4}$	50 controller-training tasks and 200 unseen test tasks per seed; 5 seeds	$p \in \{0.1, 0.2, 0.3, 0.4, 0.5\}$; residual parameter update; update scale $= 1$
OOD meta-learning	2-layer GCN; 50 epochs	SOL: 30 epochs; MAML: 100 meta-epochs; Reptile: 100 meta-epochs	50 training tasks and 200 unseen test tasks per seed; 5 seeds	$p_{\text{train}} \in \{0.1, 0.2, 0.3\}$; $p_{\text{test}} \in \{0.4, 0.5\}$; SOL $\alpha = 0.1$; 5 test-time steps for gradient-based baselines
OSM road network	GCN; 200 epochs	SOL: 50 epochs; MAML: 50 meta-training epochs; Reptile: 50 meta-training epochs	40 road-closure training tasks; 80 unseen road-closure test tasks	250 nodes, 322 edges; $c \in \{0.05, 0.10, 0.15, 0.20\}$; SOL $\alpha = 0.1$
Value adaptation	Value GCN; input dimension 3; hidden dimension 16; 50 epochs; lr $= 0.01$	MLP controller; 100 epochs in the final metric-producing run; lr $= 0.01$	5 controller-training reward tasks; 20 held-out evaluation tasks	5×5 grid; reward-sign perturbation $p \in \{0.1, 0.2, 0.3\}$

O Extended Discussion of the Second-Order Mechanism

Across the experiments, SOL instantiates the same hierarchical computation. The first-order network learns a task-dependent representation and response function, while the second-order controller operates on the state of that first-order learner to generate a task-conditioned parameter transformation when the environment changes.

The maze experiments establish this mechanism under structural perturbation: the controller improves prediction on unseen perturbed graphs, and the large-scale evaluation shows that the learned transformation transfers across tasks and perturbation strengths. The representation analyses further show that the first-order GCN develops a latent space whose relational structure is strongly aligned with the geometry of the environment.

Experiment 5 provides the complementary control. When meaningful feature–geometry correspondence is disrupted, both Pearson and Spearman distance alignment move close to zero and task performance degrades. This connects the organization of the first-order representation with the adaptation behavior of the second-order system.

The OOD meta-learning comparison characterizes a second aspect of the mechanism. SOL reaches the same high-performance regime as gradient-based meta-learning methods while applying the learned controller directly at test time, without an iterative optimization loop, test-time gradients, or labeled test examples. The OSM experiment shows that this property extends to an irregular real-world road network under structural disruption.

Experiment 7 broadens the hierarchy beyond shortest-path classification. There, the first-order learner represents a continuous value function and the second-order controller adapts it when the reward landscape changes. The resulting reductions in value-prediction error and improvement in policy accuracy show that the same second-order principle can operate across a different prediction objective and a different form of environmental change.

These results support a consistent interpretation of SOL: structured first-order representations provide the state on which the second-order learner operates, and the learned controller converts that state into rapid, task-conditioned changes of the first-order model. Across graph perturbation, OOD transfer, real-road-network adaptation, and value-function adaptation, the same hierarchical mechanism provides a common account of how learned second-order transformations support adaptation to changing environments.